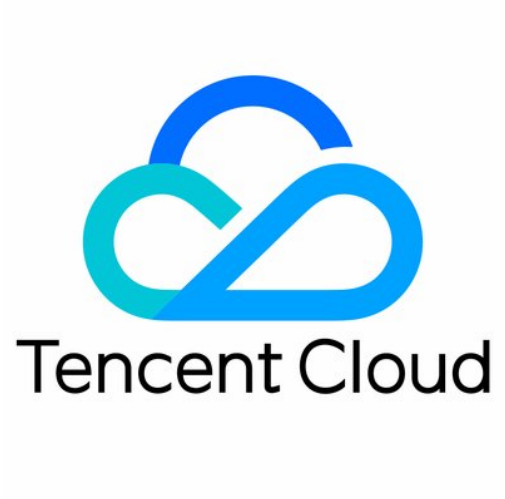
Cloud Object Storage

개발자 가이드

제품 문서



목록:

개발자 가이드
요청 생성
요청 생성 개요

최종 업데이트 날짜: : 2023-04-28 15:30:49

기본 개념

Tencent Cloud COS는 HTTP/HTTPS 프로토콜을 사용해 액세스하는 Web 스토리지 서비스입니다. REST API 또는 COS SDK를 사용해 COS에 액세스할 수 있습니다.

COS 액세스 요청 실행 시, COS 인증을 거쳐야만 리소스 작업이 가능합니다. 따라서 신분 식별 가능 여부에 근거하여, COS 액세스 요청은 익명 요청과 서명 요청의 2가지 유형으로 나뉩니다.

- 익명 요청: Authorization 또는 관련 매개변수를 가지고 있지 않거나, 관련 문자가 사용자의 신분 특징을 식별할 수 없는 경우, 익명 요청으로 간주되어 인증을 진행합니다.
- 서명 요청: 서명된 요청은 HTTP 헤더 또는 요청 패킷에 Authorization 필드가 포함돼야 합니다. 해당 필드의 콘텐츠는 Tencent Cloud의 보안 자격 증명 SecretID, SecretKey 및 요청된 일부 특징 값을 결합한 것이며, 암호화 알고리즘으로 생성됩니다.

COS SDK 호출을 사용할 경우 보안 자격 증명만 설정하면 요청을 발송할 수 있습니다. REST API 호출을 사용할 경우 서명 요청 문서를 참고하여 직접 요청 서명을 계산해야 합니다.

보안 자격 증명 얻기

CAM은 COS에 계정 및 보안 자격 증명 관련 기능과 서비스를 제공합니다. 주로 Tencent Cloud 계정에 있는 리소스의 액세스 권한을 안전하게 관리할 수 있도록 지원합니다. 사용자는 CAM을 통해 사용자(그룹)를 생성, 관리, 폐기할 수 있고 신분 관리 및 정책 관리를 사용해 기타 사용자의 Tencent Cloud 리소스 사용 권한을 제어할 수 있습니다.

루트 계정의 보안 자격 증명

루트 계정 로그인 후, CAM의 Tencent Cloud API 키 페이지를 통해 루트 계정 보안 자격 증명 SecretID와 SecretKey를 관리하고 얻을 수 있습니다. 다음은 한 그룹 키의 예시입니다.

- 36개 문자로 된 액세스 키 ID(SecretID): AKIDHZRLB9Ibhdp7Y7gyQq6BOk1997xxxxxx
- 32개 문자로 된 액세스 키(SecretKey): LYaWIuQmCSZ5ZMniUM6hiaLxHnxxxxxx

액세스 키는 고유한 계정을 표시하는 데 사용할 수 있습니다. 키 서명을 사용하여 요청을 보내면 Tencent Cloud는 요청 실행자의 신분을 식별하고 인증한 후, 신분, 리소스, 작업, 조건 등 인증을 거쳐 이 작업 실행을 허가할지 여부를 판단합니다.

> 주의 :
> 루트 계정 키는 그에 속한 모든 리소스의 모든 조작 권한을 가지고 있습니다. 키가 유출되면 클라우드 리소스 손실 가능성이 있으므로, 서브 계정을 생성하고 적절히 권한을 할당할 것을 강력히 권장합니다. 서브 계정의 키 생성을 요청하여 리소스에 액세스 및 관리하십시오.

서브 계정의 보안 자격 증명

사용자 및 클라우드 리소스를 다각도로 관리해야 하는 경우, 루트 계정에 속한 여러 개의 서브 계정을 생성한 후 리소스 권한을 여러 계정이 나누어 관리하도록 할 수 있습니다. 서브 계정 생성에 대한 자세한 내용은 CAM의 사용자 관리 관련 문서를 참고하십시오.

서브 계정을 사용한 API 요청 발송 전, 서브 계정을 위한 보안 자격 증명을 생성해야 하며, 서브 계정도 신분 식별을 위해 고유한 키 쌍을 가지고 있어야 합니다. 각 서브 계정의 사용자 정책을 작성하면 리소스에 대한 각 계정의 액세스 권한을 제어할 수 있습니다. 또한 사용자 그룹을 생성하고 사용자 그룹에 통일된 액세스 정책을 추가하면 인원 그룹별 및 리소스에 대한 통합 관리가 가능합니다.

> 주의 :
> 서브 계정은 해당 권한을 할당 받은 후, 리소스를 생성 또는 수정할 수 있습니다. 이 경우 리소스는 여전히 루트 계정에 속하며, 해당 리소스에서 발생하는 비용 역시 루트 계정이 지불합니다.

임시 보안 자격 증명

루트 계정 또는 서브 계정의 보안 자격 증명을 통한 리소스 액세스 외에도 Tencent Cloud는 역할 생성을 지원하고, 임시 보안 자격 증명을 사용하여 Tencent Cloud 리소스를 관리합니다. 역할 관련 기본 개념 및 사용 방법은 CAM의 역할 관리 문서를 참고하십시오.

역할은 가상 신분이기 때문에 영구적인 키를 가지고 있지 않으므로 Tencent Cloud의 CAM은 임시 보안 자격 증명을 생성하기 위한 STS API를 제공합니다.
사용 방법과 세부 예시는 Using Role 문서나 CreateRole 문서에서 임시 보안 자격 증명 생성 방식을 참고하십시오. 임시 보안 자격 증명에는 보통 **정책 제한**(작업, 리소스, 조건) 및 **시간 제한**(시작 및 종료 유효 시간)만 포함되므로, 생성된 임시 보안 자격 증명은 자체 배포하거나 직접 사용할 수 있습니다.

임시 보안 자격 증명을 생성하는 인터페이스를 호출하면 한 쌍의 임시 키(tmpSecretId/tmpSecretKey)와 보안 토큰(sessionToken)을 획득하게 되며, 이는 COS에 액세스하는 데 사용하는 보안 자격 증명을 구성합니다. 예시는 다음과

같습니다.

- 41개 문자로 된 보안 토큰(SecurityToken): 5e776c4216ff4d31a7c74fe194a978a3ff2xxxxxx
- 36개 문자로 된 임시 액세스 키 ID(SecretID): AKIDcAZnqgar9ByWq6m7ucIn8LNEuYxxxxxx
- 32개 문자로 된 임시 액세스 키(SecretKey): VpxrX0IMCpHXWL0Wr3KQNCqJixxxxxxx

해당 인터페이스는 또한 `expiration` 필드를 통해 임시 보안 자격 증명의 유효 시간을 반환하는데, 이는 이 시간 내에 해당 보안 자격 증명을 사용해야만 요청을 실행할 수 있음을 의미합니다.

Tencent Cloud COS는 임시 키 생성에 사용되는 간단한 서버 SDK를 제공하며, COS STS SDK에 액세스하여 획득할 수 있습니다. 임시 보안 자격 증명을 받은 후 REST API를 사용하여 요청을 발송하려면 HTTP 헤더 또는 POST 요청 패킷의 form-data에 `x-cos-security-token` 필드를 전송하여 해당 요청에 사용한 보안 토큰을 표시한 한 후, 임시 액세스 키 쌍을 사용하여 요청 서명을 계산해야 합니다. COS SDK를 사용한 액세스 실행은 각 SDK 문서의 관련 부분을 참고하십시오.

도메인 이름 액세스

REST API

리전 및 액세스 도메인 문서에 REST API 호출 실행에 사용하는 리전 리스트가 나열되어 있습니다.

COS는 가상 호스팅형 도메인을 사용하여 버킷에 액세스하는 것을 권장합니다. HTTP 요청 실행 시 `<bucketname-appid>.cos.<region>.myqcloud.com` 과 같이 직접 `Host` 헤더를 통해 액세스해야 하는 버킷을 가져옵니다. 가상 호스팅형 도메인을 사용해 가상 서버의 [루트 디렉터리]와 유사한 기능을 구현하며, favicon.ico, robots.txt, crossdomain.xml과 같은 파일을 호스팅하는 데 사용할 수 있습니다. 이 파일들은 여러 응용 프로그램이 호스팅 웹 사이트를 식별할 때 기본적으로 가상 서버의 [루트 디렉터리]에서 콘텐츠를 인덱스합니다.

경로형 요청을 통해 버킷에 액세스할 수도 있습니다. 예를 들어, `cos.<region>.myqcloud.com/<bucketname-appid>/` 의 경로를 사용하여 버킷에 액세스할 수 있습니다. 이와 마찬가지로, Host와 서명 요청은 `cos.<region>.myqcloud.com` 을 사용해야 하며, SDK는 기본적으로 해당 액세스 방식을 지원하지 않습니다.

정적 웹 사이트 도메인

버킷의 정적 웹 사이트 기능 활성화 시, 관련 기능을 사용할 수 있도록 가상 호스팅형 도메인을 할당합니다. 정적 웹 사이트 도메인의 성능은 REST API와 다릅니다. 정적 웹 사이트 도메인은 특정 인덱스 페이지, 오류 페이지 및 리디렉션 설정을 제외하고 GET/HEAD/OPTIONS Object 등 몇 가지 작업만 지원하며, 업로드 또는 리소스 설정 작업은 지원하지 않습니다.

정적 웹 사이트 도메인 예시로는 `<bucketname-appid>.cos-website.<region>.myqcloud.com` 가 있고, 콘솔의 버킷 [기본 설정 - 정적 웹 사이트 설정] 모듈을 통해 이 도메인을 얻을 수 있습니다.

사설망 액세스

Tencent Cloud COS의 액세스 도메인은 스마트 DNS 레졸루션을 사용하여 각 통신사 환경의 인터넷에서 COS 액세스를 검증하고 최적의 링크를 제공합니다.

Tencent Cloud에서 배포한 CVM 서비스가 사설망에서 COS에 액세스하려면, 먼저 CVM과 COS 버킷이 동일한 리전에 속하는지 확인한 후 CVM에서 `nslookup` 명령을 사용하여 COS 도메인 이름을 해석해야 합니다. 내부 IP가 반환되면 CVM과 COS 간에 사설망 액세스를 나타내고, 그렇지 않으면 공중망 액세스를 나타냅니다.

Tencent Cloud에서 배포한 CVM 서비스의 리전이 COS 버킷이 속한 리전과 다르지만, COS 사용 가능 리전 범위 내에 있는 경우, COS 사설망 전역 가속 도메인 이름을 통해 파일에 액세스하여 CVM과 COS의 리전 간 액세스를 구현할 수 있습니다.

사설망 액세스 판단 방법

동일한 리전 내의 Tencent Cloud 제품은 사설망을 통해 서로 액세스할 수 있으므로 트래픽 비용이 발생하지 않습니다. 따라서 비용을 절약하기 위해 다른 Tencent Cloud 제품을 구매할 때 동일한 리전을 선택하는 것이 좋습니다.

> 주의 :
> 퍼블릭 클라우드 리전의 사설망은 금융 클라우드 리전의 사설망과 상호 연결되지 않습니다.

다음은 사설망을 통한 액세스를 확인하는 방법입니다.

Tencent CVM을 통한 COS 액세스의 경우, 내부 인터넷을 이용하여 COS에 액세스했는지 판단하려면, CVM에서 `nslookup` 명령어를 사용하여 COS 도메인을 해석할 수 있습니다. 내부 IP를 반환한다면 CVM과 COS 간 사설망으로 액세스한 것이고, 아니라면 공중망으로 액세스한 것입니다.

> 설명 :
> 내부 IP 주소는 일반적으로 `10.*.*.*` , `100.*.*.*` 형식이며, VPC 네트워크는 일반적으로 `169.254.*.*` 등의 형식입니다. 해당 두 가지 형식의 IP는 사설망에 속합니다.

`examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com` 이 타깃 버킷 주소라고 가정하면, `nslookup` 명령어 실행 후 다음과 같은 정보를 확인할 수 있습니다.

```
nslookup examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
Server:   10.138.224.65
Address:  10.138.224.65  #53
Name:     examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
Address:  10.148.214.13
Name:     examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
Address:  10.148.214.14
```

여기서 `10.148.214.13` 와 `10.148.214.14` 의 두 IP는 사설망을 통해 COS에 액세스했음을 의미합니다.

연결성 테스트

기본 연결 테스트

COS는 HTTP 프로토콜을 사용하여 외부로 서비스를 제공합니다. 가장 기본적인 `telnet` 툴로 COS 액세스 도메인의 80 포트에 대해 액세스 테스트를 실행할 수 있습니다.

공중망을 통한 액세스 예시는 다음과 같습니다.

```
telnet examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com 80
Trying 14.119.113.22...
Connected to gz.file.myqcloud.com.
Escape character is '^]'.
```

같은 리전 내 Tencent Cloud CVM(기본 네트워크) 액세스 예시는 다음과 같습니다.

```
telnet examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com 80
Trying 10.148.214.14...
Connected to 10.148.214.14.
Escape character is '^]'.
```

같은 리전 내 Tencent Cloud CVM(VPC 네트워크) 액세스 예시는 다음과 같습니다.

```
telnet examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com 80
Trying 169.254.0.47....
Connected to 169.254.0.47.
Escape character is '^]'.
```

액세스 환경에 관계 없이, 명령어가 `Escape character is '^]'.` 필드를 반환한다면 연결에 성공했음을 의미합니다.

인터넷 테스트를 통한 설명

인터넷을 통해 COS에 액세스하면 ISP 네트워크를 거치게 됩니다. ISP 네트워크는 ICMP 프로토콜의 `ping` 또는 `traceroute` 등 툴을 사용한 연결성 테스트를 허용하지 않습니다. 따라서 TCP 프로토콜의 툴을 사용한 연결성 테스트를 권장합니다.

> 주의 :
> 인터넷을 통해 액세스하면 다양한 인터넷 환경의 영향을 받을 수 있습니다. 만약 액세스가 원활하지 못하다면, 로컬 네트워크 링크를 진단하거나 로컬 통신사에 연락하여 피드백을 받으십시오.

ISP가 ICMP 프로토콜을 허용하는 경우 `ping` , `traceroute` 또는 `mtr` 툴을 사용해 링크 상황을 검사할 수 있습니다. ISP가 ICMP 프로토콜을 허용하지 않을 경우 `psping` (Windows 환경, Microsoft 공식 홈페이지에서 다운로드) 또는 `tcping` (크로스 플랫폼 소프트웨어) 등의 툴을 사용해 딜레이 시간을 테스트할 수 있습니다.

사설망 테스트 설명

같은 리전 내 Tencent Cloud VPC 네트워크를 통해 객체 스토리지 COS에 액세스하면, ICMP 프로토콜의 `ping` 또는 `traceroute` 등 툴을 사용하여 연결성을 테스트할 수 없습니다. 기본 연결 테스트 중 `telnet` 명령어를 사용하여 테스트를 진행할 것을 권장합니다.

`psping` 또는 `tcping` 등의 툴을 사용하여 직접 액세스 도메인의 80 포트에 대한 딜레이 시간 테스트를 시도해 볼 수도 있습니다. 테스트 전, `nslookup` 명령어를 통해 액세스 도메인이 사설망 주소에 정확하게 해석되었는지 조회 및 확인하십시오.

버킷
버킷 개요

최종 업데이트 날짜: : 2023-03-14 14:46:39

소개

버킷(Bucket)은 객체를 저장하는 '컨테이너'로 이해할 수 있으며, 이 '컨테이너'는 용량의 상한이 없습니다. 객체는 폴더와 디렉터리의 개념이 없는 평면 구조의 버킷에 저장됩니다. 하나 또는 여러 버킷에 객체를 저장하도록 선택할 수 있습니다.

> 설명 :
> 버킷은 객체를 얼마든지 포함할 수 있지만 하나의 루트 계정은 최대 200개의 버킷만 생성할 수 있습니다.

버킷 이름 생성 규칙

버킷 이름은 하이픈 '-'으로 연결된 버킷 이름(BucketName)과 APPID로 구성됩니다. 예를 들어 버킷 이름 'examplebucket-1250000000'에서 examplebucket은 사용자 정의 문자열이고 1250000000은 시스템 생성 숫자 문자열(APPID)입니다. API 및 SDK 예시에서 버킷의 이름은 `<bucketname-appid>` 의 형식으로 생성됩니다.

- 버킷 이름(BucketName): 사용자가 수동으로 입력한 문자열로 이뤄지며, 이름 생성 규칙은 다음과 같습니다.
- 알파벳 소문자와 숫자 [a-z, 0-9], 하이픈 '-' 및 그 조합만 지원합니다.
- 버킷 이름에 포함될 수 있는 문자 수는 **리전 약칭** 및 **APPID**의 길이로 제한됩니다. 결합된 도메인 이름은 최대 60자까지 가능합니다. 예를 들어 도메인 이름 `1234567890123456789012-1250000000.cos.ap-beijing.myqcloud.com` 은 60자를 포함합니다.
- 버킷 이름 생성 시, '-'으로 시작하거나 끝날 수 없습니다.
- APPID는 Tencent Cloud 계정 신청 완료 후 받은 계정입니다. 시스템에서 자동으로 할당되는 고유한 계정이며 계정 정보에서 조회할 수 있습니다. 콘솔을 통해 버킷 생성 시 사용자를 입력할 필요가 없고, 툴과 API, SDK를 사용할 경우 APPID를 지정해야 합니다.

아래는 효율적인 버킷 이름 생성 예시입니다.

- examplebucket-1-1250000000
- mybucket123-1250000000
- 1-newproject-1250000000

버킷 소속 리전

리전(Region)은 객체 스토리지 COS의 데이터센터가 있는 지역을 의미합니다. 객체 스토리지는 사용자가 다른 리전에 버킷을 생성하는 것을 허용합니다. 업무 지역에서 가장 가까운 리전을 선택하고 버킷을 생성함으로써, 딜레이와 비용을 줄일 수 있고 컴플라이언스 기준을 충족할 수 있습니다.

예를 들어, 업무가 화난 지역에 분포되어 있는 경우 광저우 리전을 선택해 버킷을 생성하면 객체의 업로드 및 다운로드 속도를 향상시킬 수 있습니다. 리전에 관한 자세한 내용은 리전 및 액세스 도메인 문서를 참고하십시오.

주의 :
리전은 버킷 생성 시 반드시 지정해야 하며, 지정 후 변경할 수 없습니다. 해당 버킷의 모든 객체는 상응하는 데이터센터에 저장되며, 객체 레벨별 리전 설정은 아직 지원하지 않습니다.

권한 유형

버킷은 기본적으로 공개 권한과 사용자 권한의 두 가지 권한을 제공합니다.

설명 :

- 버킷이 개인 읽기/쓰기이거나 지정된 계정에 사용자 권한이 부여된 경우 객체 요청 시 본인 확인을 위해 서명이 필요하며 서명 방법은 Request Signature를 참고하십시오.
- 버킷이 공개 읽기/개인 쓰기 또는 공개 읽기/쓰기인 경우 객체 요청 시 서명이 필요하지 않아, 익명 사용자가 링크를 통해 객체에 직접 액세스할 수 있는 리스크가 있습니다. 데이터가 유출될 리스크가 있으니 신중하게 설정하십시오.

공개 권한

공개 권한에는 개인 읽기/쓰기, 공개 읽기/개인 쓰기 및 공개 읽기/쓰기가 포함됩니다. COS 콘솔에서 버킷의 **권한 관리**에서 버킷 액세스 권한을 수정할 수 있습니다. 자세한 내용은 액세스 제어 기본 개념을 참고하십시오.

- 개인 읽기/쓰기
  해당 버킷의 생성자 및 권한 보유 계정만 해당 버킷에 있는 객체에 대한 읽기/쓰기 권한을 가지며, 그 외 사용자는 해당 버킷에 있는 객체에 대한 읽기/쓰기 권한이 없습니다. 버킷 액세스의 기본 권한은 개인 읽기/쓰기이므로 사용을 권장합니다.
- 공개 읽기와 개인 쓰기
  익명 방문자를 포함해 누구나 해당 버킷의 객체에 대한 읽기 권한을 가지지만, 해당 버킷의 객체에 대한 쓰기 권한은 버킷 생성자 및 권한 보유 계정만이 가집니다.

- 공개 읽기/쓰기
  익명 방문자를 포함해 누구나 해당 버킷의 객체에 대한 읽기/쓰기 권한을 가지므로 사용을 권장하지 않습니다.

사용자 권한

루트 계정은 기본적으로 버킷에 대한 모든 권한 즉, 전체 제어 권한을 갖습니다. 그 외에, COS는 서브 계정을 추가 생성하여 데이터 읽기/쓰기, 권한 읽기/쓰기, 심지어 완전 통제 가능한 최고 권한을 갖도록 지원합니다.

버킷 작업

사용자는 Tencent Cloud 콘솔, 툴, API, SDK 등 다양한 방식으로 버킷 및 설정 속성을 관리할 수 있습니다. 예를 들어 버킷 설정은 정적 웹 사이트 호스팅 및 버킷의 액세스 권한 설정에 사용됩니다. 아래 예시는 일부 기능에 대한 설정 가이드이며, 버킷 기능 설정에 대한 자세한 내용은 버킷 개요를 참고하십시오.

- 버킷 생성
- 정적 웹 사이트 설정
- 액세스 권한 설정
- 링크 도용 방지 설정

관련 설명

- 객체 스토리지는 수평적 구조로 객체를 저장하며 폴더 개념이 없습니다. 자세한 내용은 객체 개요 문서의 '폴더 및 목록' 부분을 참고하십시오.
- 하나의 루트 계정(즉, 동일한 APPID)은 리전 구분 없이 최대 200개의 버킷을 생성할 수 있으나, 버킷 내 객체 수량에는 제한이 없습니다.
- Tencent Cloud COS에서 동일한 APPID의 버킷은 고유한 이름을 가지며 이름을 변경할 수 없습니다.
- 버킷 생성 완료 후에는 버킷 이름을 변경할 수 없고, 버킷을 삭제한 후 다시 생성하면 이름을 변경할 수 있습니다.
- 리전 설정 완료 후에는 리전을 수정할 수 없으므로, 사용자는 버킷 생성 시 소속 리전을 확인해야 합니다.

버킷 생성

최종 업데이트 날짜: : 2021-11-01 14:46:33

파일을 Cloud Object Storage(COS)에 보관할 경우, 우선 객체를 보관할 버킷을 생성해야 합니다. 콘솔, 툴, API 또는 SDK를 통해 버킷을 생성할 수 있습니다.

버킷 생성 후, 객체를 해당 버킷에 업로드할 수 있으며, 버킷을 위한 기타 기능을 설정할 수 있습니다. 예를 들어, 정적 웹 사이트 설정, 버킷 태그 설정, 버킷 암호화 설정 등이 있습니다. 가이드 설정 관련 자세한 내용은 콘솔 개요를 참고하십시오.

제한 설명

1. 하나의 루트 계정은 최대 200개의 버킷을 생성할 수 있습니다.
2. 버킷 생성 완료 후에는 버킷 이름과 소속 리전을 수정할 수 없습니다.
3. 하나의 루트 계정에서 각 버킷은 고유한 이름을 가지며 이름을 변경할 수 없습니다.
4. 버킷 이름은 알파벳 소문자와 숫자, 즉 [a-z, 0-9], 하이픈 '-' 만 사용 가능합니다. 버킷 이름의 최대 글자 수는 **리전 약칭** 및 **APPID** 글자 수의 영향을 받습니다. 완전한 요청 도메인의 최대 글자 수는 60자입니다. 예를 들어 요청 도메인 `12345678901234567890-1250000000.cos.ap-beijing.myqcloud.com` 은 총 60자입니다.
5. 버킷 이름 생성 시, '-'으로 시작하거나 끝날 수 없습니다.

사용 방법

COS 콘솔 사용

COS 콘솔을 사용하여 버킷을 생성할 수 있습니다. 자세한 내용은 버킷 생성 콘솔 문서를 참고하십시오.

툴 사용

COSBrowser, COSCMD 등의 툴을 사용하여 버킷을 생성할 수 있습니다. 자세한 내용은 툴 개요를 참고하십시오.

REST API 사용

REST API를 사용하여 버킷 생성 요청을 실행할 수 있습니다. 자세한 내용은 PUT Bucket API 문서를 참고하십시오.

SDK 사용

SDK 버킷 생성 메소드를 호출할 수 있습니다. 자세한 내용은 아래 각 언어로 된 SDK 문서를 참고하십시오.

- Android SDK
- C SDK

- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK
- Node.js SDK
- PHP SDK
- Python SDK
- 미니프로그램 SDK

버킷 삭제

최종 업데이트 날짜: : 2023-03-23 15:49:28

임의 상황에서 버킷을 삭제해야 할 경우, 콘솔, 툴, API 또는 SDK 방식으로 버킷을 삭제할 수 있습니다.

> 주의 :
> 버킷을 삭제한 후에는 복구할 수 없으니 신중히 하시기 바랍니다.

제한 설명

- 현재는 데이터가 이미 비어 있는 버킷 삭제만 지원합니다. 버킷에 객체나 파일 조각이 여전히 있을 경우 삭제에 실패합니다. 버킷 삭제 실행 전에 버킷 내부에 객체가 비어 있는지 확인하십시오. 자세한 내용은 버킷 비우기에서 알아보십시오.
- 버킷 삭제 시, 작업자가 해당 작업에 대한 권한을 보유했는지, 정확한 버킷 이름(Bucket)과 리전(Region) 매개변수가 전달되었는지 확인해야 합니다.

사용 방법

COS 콘솔 사용

객체 스토리지 콘솔을 사용하여 버킷을 삭제할 수 있습니다. 자세한 내용은 버킷 삭제 콘솔 가이드 문서를 참조하십시오.

사용 툴

COSBrowser, COSCMD 등의 툴을 사용하여 버킷을 삭제할 수 있습니다. 자세한 내용은 툴 개요를 참조하십시오.

REST API 사용

REST API를 사용하여 버킷 삭제 요청을 실행할 수 있습니다. 자세한 내용은 DELETE Bucket API 문서를 참조하십시오.

SDK 사용

SDK의 버킷 삭제 메서드를 호출할 수 있습니다. 자세한 내용은 아래 각 언어로 된 SDK 문서를 참조하십시오.

- Android SDK
- C SDK

- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- Mini Program SDK

객체
객체 개요

최종 업데이트 날짜: : 2022-09-28 12:04:32

소개

객체(Object)는 Cloud Object Storage(COS)의 기본 단위로서 이미지, 문서, 멀티미디어 파일 등 다양한 포맷의 데이터라고 할 수 있습니다. 버킷(Bucket)은 객체의 저장 장치이며 각 버킷은 용량 제한 없이 객체를 수용할 수 있습니다. 객체는 COS에서 다양한 스토리지 유형으로 지정 가능합니다. 자세한 내용은 스토리지 유형 개요을 참고하십시오.

각 객체는 객체 키(ObjectKey), 객체 값(Value), 객체 메타데이터(Metadata)로 구성됩니다.

- 객체 키(ObjectKey): 객체 키는 버킷에 있는 객체의 고유 ID이며 일반적으로 파일 경로로 이해할 수 있습니다. API, SDK 예시 중 객체의 이름 생성 포맷은 `<objectkey>` 입니다.
- 객체 값(Value): 업로드하는 객체 자체를 뜻하며 일반적으로 파일 콘텐츠(Object Content)로 이해할 수 있습니다.
- 객체 메타데이터(Metadata): 파일 수정 시간, 스토리지 유형과 같은 파일 속성에 해당하는 키 값 쌍입니다. 객체 업로드 후 조회가 가능합니다.

객체 키

Tencent Cloud COS 객체는 합법적인 객체 키를 보유해야 하며 객체 키(ObjectKey)는 버킷에 있는 객체의 고유 ID입니다.
예를 들어, 객체의 액세스 주소 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/folder/picture.jpg` 에서 객체 키는 `folder/picture.jpg` 입니다.

이름 생성 규칙

- 객체 키 이름 생성 시 모든 UTF-8 문자를 사용할 수 있습니다. 이름과 기타 응용 프로그램의 호환성을 위해 영어 알파벳 대소문자와 숫자 [a-z, A-Z, 0-9]의 조합을 사용하는 것이 좋습니다.
- 코드 길이는 최대 850바이트입니다.
- 이름은 슬래시 `/` 또는 백슬래시 `\` 로 시작할 수 없습니다.
- 객체 키에는 ASCII 제어 부호인 위(↑), 아래(↓), 오른쪽(→), 왼쪽(←)을 포함할 수 없으며 이는 각각 CAN(24), EM(25), SUB(26), ESC(27)에 대응됩니다.
- 파일 이름에 `*` 및 `%` 와 같은 특수 기호를 사용하지 마십시오.

설명 :

사용자가 업로드한 파일 또는 폴더 이름에 중국어 문자가 포함되어 있는 경우 파일 또는 폴더에 액세스하거나 요청할 때 중국어 문자는 URL Encode 규칙에 따라 백분율 인코딩된 문자열로 변환됩니다.
예를 들어, `문서.doc` 에 액세스할 때 객체 키는 `문서.doc` 이지만 실제로는 URL Encode 규칙에 따라 백분율 인코딩된 문자열은 `%e6%96%87%e6%a1%a3.doc` 입니다.

다음은 유효한 객체 키 이름의 예시입니다.

- doc/exampleobject
- my.great_photos-2016/01/me.jpg
- videos/2016/birthday/video.wmv

특수 부호

어떤 부호는 객체 키에서 16진법 형태로 URL 인코딩 또는 참조 테이블이 필요합니다. 그 중 일부는 출력이 불가능하여 브라우저가 처리할 수 없으므로 특수한 프로세스가 필요한데, 해당 부호는 다음과 같습니다.

,	:	;	=
&	$	@	+
?	ASCII 부호 범위: 00-1F 16진법(0-31 10진법)과 7F(127 10진법)	(빈칸)	

또한 어떤 부호는 많은 특수한 프로세스를 거쳐야만 모든 응용 프로그램 간 일관성을 유지할 수 있으므로 바로 사용하지 않는 것을 권장합니다. 사용을 피해야 할 부호는 아래와 같습니다.

`	^	"	\
{	}	[	]
~	%	#	|
*	>	<	ASCII 128-255 10진법

객체 액세스 주소

객체 액세스 주소는 버킷 액세스 주소와 객체 키로 구성되어 `<버킷 도메인>/<객체 키>` 의 구조로 이루어져 있습니다.
예를 들어, 객체 `exampleobject.txt` 를 광저우(화남) 리전 버킷 `examplebucket-1250000000` 에 업로드하면 `exampleobject.txt` 액세스 주소는 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/exampleobject.txt` 입니다.

폴더와 디렉터리

COS 자체에는 폴더나 디렉터리의 개념이 없습니다. COS는 객체 `project/a.txt` 를 업로드했다고 해서 `project` 폴더를 생성하지는 않습니다. 간편하게 사용할 수 있도록 COS는 콘솔과 COS browser 등의 도식화 툴에서 '폴더' 또는 '디렉터리' 디스플레이 방식을 모방합니다. 예를 들어, 키 값을 `project/` 로 하여 생성하면 콘텐츠가 없는 객체는 일반적인 폴더로 표시됩니다.

예를 들어 API, SDK를 통해 객체 `project/doc/a.txt` 를 업로드하면 세퍼레이터 `/` 가 '폴더'의 디스플레이 방식을 모방하고, 콘솔에서 '폴더' `project` 와 `doc` 가 나타납니다. 그 중 `doc` 는 `project` 의 하위 '폴더'이며 `a.txt` 파일을 포함합니다.

> 주의 :
> 버킷의 각 객체는 각 분산형 클러스터에 수평적으로 분포되어 있습니다. 따라서 특정한 객체 키 접두사로 객체의 크기를 바로 알 수 없으며 각 객체 크기를 누적 계산해야 전체 크기를 알 수 있습니다.

파일이나 디렉터리 삭제 작업은 비교적 복잡합니다. 자세한 내용은 다음과 같습니다.

경로 삭제	객체 또는 폴더 삭제	결과
콘솔	폴더 `project`	객체 키 접두사가 `project/` 인 모든 객체를 삭제합니다.
콘솔	객체 `project/doc/a.txt`	`project` 와 `doc` 폴더는 보관할 수 있습니다.
API, SDK	객체 `project/` 또는 `project/doc/`	객체 `project/doc/a.txt` 는 보관할 수 있습니다. 폴더 내 객체를 모두 삭제하고 싶다면 코드 순회를 이용해 폴더 내 객체를 삭제해야 합니다.

객체 메타데이터

객체 메타데이터는 객체의 키 값 쌍을 의미하며, 서버가 HTTP 프로토콜 형식으로 HTML 데이터를 브라우저에 전송하기 전 보내는 문자열로써, HTTP Header라고도 합니다. 객체 업로드 시 HTTP Header를 수정하여 페이지 응답 형식을 바꾸거나 캐시 수정 시간과 같은 설정 정보를 전송할 수 있습니다.

객체 메타데이터는 시스템 메타데이터와 사용자 정의 메타데이터의 2가지 종류가 있습니다.

> 설명 :
> 객체의 HTTP Header를 수정해도 객체 자체를 수정할 수는 없습니다.

시스템 메타데이터

객체 수정 시간, 객체 크기, 스토리지 유형 등 객체의 속성 정보를 의미합니다.

이름	설명
Content-Length	RFC 2616에서 정의한 HTTP 요청 콘텐츠 길이(바이트)를 의미합니다.
Last-Modified	객체 생성 날짜 또는 마지막 수정 날짜(나중에 생성된 객체 기준)를 의미합니다.
x-cos-version-id	객체 버전 ID. 버킷 버전 제어가 활성화될 경우 객체의 버전 ID를 반환하여 ID가 동일한 객체의 이전 버전을 표시합니다.
ETag	PUT Object를 통해 업로드한 객체의 MD5 값이나 멀티파트 업로드나 이전 버전 API를 통해 업로드한 객체의 고유 ID로 인증 기능이 없습니다.

사용자 정의 메타데이터

Content-Type, Cache-Control, Expires, x-cos-meta-* 등 객체의 사용자 정의가 가능한 매개변수를 의미합니다. 자세한 내용은 사용자 정의 Headers를 참조하십시오.

이름	설명
Cache-Control	RFC 2616에서 정의한 캐시 정책으로, 객체 메타데이터로 저장됩니다.
Content-Disposition, Content-Encoding, Content-Type	RFC 2616에서 정의한 파일 이름/인코딩 포맷/콘텐츠 유형(MIME)으로, 객체 메타데이터로 저장됩니다.
Expires	RFC 2616에서 정의한 유효하지 않은 캐시 시간으로, 객체 메타데이터로 저장됩니다.
x-cos-acl	객체의 액세스 제어 리스트(ACL) 속성을 정의합니다. 유효 값은 private, public-read-write, public-read이고, 기본값은 private입니다.
x-cos-grant-*	권한을 보유한 계정에 특정 권한을 부여합니다.
x-cos-meta-*	사용자 정의한 헤더 정보를 허용하며, 객체 메타데이터로 반환합니다. (용량 제한 2KB)
x-cos-storage-class	객체의 스토리지 레벨을 설정하며, 열거 값은 STANDARD, STANDARD_IA, ARCHIVE이고, 기본값은 STANDARD입니다.
x-cos-server-side-encryption	객체를 위한 서버 암호화 활성화 여부와 암호화 방식을 지정합니다.

객체 작업

사용자는 Tencent Cloud 콘솔, 툴, API, SDK 등의 방식으로 객체를 관리할 수 있습니다.

> 주의 :
> 지원하는 최대 업로드 파일 크기에 따라 간편 업로드와 멀티파트 업로드로 나눌 수 있습니다.
>
> - 간단 업로드는 객체 크기 5GB 이내로 제한됩니다.
> - 멀티파트 업로드 사용 시 각 파트의 크기는 5GB 이내로 제한되고 파트 수는 10,000개보다 작아야 합니다. 즉 최대 업로드 객체 크기는 약 48.82TB입니다. 제한 사항에 대한 자세한 내용은 규격 및 제한을 참조하십시오.

객체 업로드 완료 후 사용자는 콘솔에서 객체 관련 설정을 진행할 수 있습니다. 자세한 사항은 아래 내용을 참조하십시오.

- 객체 검색
- 객체 정보 조회
- 객체의 액세스 권한 설정
- 사용자 정의 Headers

객체 서브 리소스

COS는 버킷 및 객체와 관련된 서브 리소스를 가집니다. 서브 리소스는 객체에 종속되므로 독립적으로 존재할 수 없고, 객체 또는 버킷과 같은 다른 엔티티와 연결됩니다. 액세스 제어 리스트(Access Control List)는 특정 객체의 액세스 제어 정보 리스트를 의미하며 COS에서 객체의 서브 리소스 역할을 합니다.

액세스 제어 리스트는 권한을 보유한 계정과 허가된 권한 리스트로 나눌 수 있고, 이를 통해 객체의 액세스 제어를 관리합니다. 객체 생성 시 ACL은 객체를 전체 제어할 수 있는 객체 소유자를 식별합니다. 사용자는 객체 ACL을 인덱스하거나 업데이트된 권한 리스트로 교체할 수 있습니다.

> 설명 :
> ACL을 업데이트하려면 기존 ACL을 교체해야 합니다.

액세스 권한 유형

COS는 **공개 권한**과**사용자 권한**이라는 2가지 객체 권한 설정 유형을 지원합니다.
공개 권한은 권한 상속, 개인 읽기/쓰기와 공개 읽기/쓰기를 포함합니다.

- 권한 상속: 객체가 버킷 권한을 상속하여 버킷과 액세스 권한이 일치합니다. 객체에 액세스할 경우 COS가 객체 권한을 버킷 권한 상속으로 인식하면 버킷과 권한을 매칭하여 액세스에 응답합니다. 다른 새로운 객체를 추가할 경우 버킷 권한의 상속을 기본으로 합니다.
- 개인 읽기/쓰기: 객체에 액세스할 경우 COS가 객체 권한을 개인 읽기/쓰기로 인식하면 버킷의 권한 종류와 무관하게 Request Signature를 통해서 액세스해야 합니다.
- 공개 읽기/개인 쓰기: 객체에 액세스할 경우 COS가 객체 권한을 공개 읽기로 인식하면 버킷의 권한 종류와 무관하게 객체를 바로 다운로드할 수 있습니다.

사용자 권한: 루트 계정은 기본적으로 객체에 대한 모든 권한 즉, 전체 제어 권한을 갖습니다. 그 외에, COS는 서브 계정을 추가 생성하여 데이터 읽기/쓰기, 권한 읽기/쓰기 및 전체 제어가 가능한 최고 권한을 갖도록 지원합니다.

적용 시나리오

개인 읽기/쓰기가 가능한 버킷에서 특정 객체에 공개 액세스를 설정했거나, 공개 읽기/쓰기가 가능한 버킷에서 특정 객체에 필요한 인증을 설정한 경우에만 액세스가 가능합니다.

스토리지 유형
스토리지 유형 개요

최종 업데이트 날짜: : 2023-05-24 15:45:08

COS(Cloud Object Storage)는 액세스 빈도 및 재해 복구 수준이 다른 객체에 대해 MAZ_STANDARD, MAZ_STANDARD_IA, MAZ_INTELLIGENT TIERING, INTELLIGENT TIERING, STANDARD, STANDARD_IA, ARCHIVE 및 DEEP ARCHIVE 스토리지 클래스를 제공합니다. 활성 객체가 COS에 있는 방법을 나타내며 액세스 빈도, 내구성, 가용성, 대기 시간 등이 서로 다릅니다. 필요에 따라 객체를 업로드할 스토리지 클래스를 선택할 수 있습니다.

주의 :

- 객체를 업로드할 때, 스토리지 유형을 설정하지 않으면 기본적으로 **STANDARD**에 업로드됩니다.
- 다중 AZ에 대한 자세한 내용은 Overview of Multi-AZ Feature를 참고하십시오.

MAZ_STANDARD/STANDARD

MAZ_STANDARD 및 STANDARD 스토리지 클래스는 모두 핫 데이터용으로 설계된 매우 안정적이고 사용 가능하며 강력한 객체 스토리지 서비스이며 낮은 대기 시간과 높은 처리량을 특징으로 합니다.

MAZ_STANDARD는 STANDARD보다 데이터 내구성과 서비스 가용성이 높습니다. 동일한 리전의 다른 데이터 센터에 데이터를 저장하기 위해 다른 스토리지 메커니즘을 사용하여 한 데이터 센터의 장애가 전체 서비스에 영향을 미치지 않도록 방지하고 비즈니스 안정성을 더욱 보장합니다.

적용 시나리오

인기 동영상, 소셜 이미지, 모바일 앱, 게임 프로그램 및 정적 웹사이트를 포함하여 많은 핫스팟 파일 또는 빈번한 데이터 액세스와 관련된 시나리오에 적합합니다.

STANDARD 스토리지 클래스는 일반적인 사용을 위해 설계되었으며 대부분의 사용 사례를 다룹니다. MAZ_STANDARD보다 비용 효율적입니다.

그러나 MAZ_STANDARD는 데이터 내구성과 서비스 가용성이 더 높기 때문에 키 파일, 상업 데이터 및 민감한 정보를 포함하여 요구 사항이 더 높은 비즈니스 시나리오에 적합합니다.

MAZ_STANDARD_IA/STANDARD_IA

MAZ_STANDARD_IA 및 STANDARD_IA 스토리지 클래스는 스토리지 비용과 액세스 대기 시간이 낮은 매우 안정적인 객체 스토리지 서비스입니다. 저렴한 가격으로 밀리초 단위로 첫 번째 바이트에 액세스할 수 있으므로 기다리지 않고 신속하게 데이터를 검색할 수 있습니다. STANDARD와 달리 데이터에 액세스할 때 데이터 검색 요금이 부과됩니다.

MAZ_STANDARD_IA는 STANDARD_IA와 다른 스토리지 메커니즘을 사용하여 동일한 리전의 다른 데이터 센터에 데이터를 저장하여 한 데이터 센터의 장애가 전체 서비스에 영향을 미치지 않도록 방지하고 비즈니스 안정성을 한층 더 보장합니다.

적용 시나리오

클라우드 디스크 데이터, 빅데이터 분석, 정부 및 기업 데이터, 빈도가 낮은 아카이브, 모니터링 데이터 등 액세스 빈도가 낮은(예: 월 1 - 2회) 시나리오에 적합합니다.

> 주의 :
> MAZ_STANDARD_IA 및 STANDARD_IA 모두 최소 스토리지 요구 사항이 있습니다. 보관 기간이 30일 미만인 경우 청구서는 30일로 계산됩니다. 마찬가지로 파일 크기가 64KB 미만인 경우 요금은 64KB로 계산됩니다(파일 크기가 64KB보다 크거나 같으면 실제 파일 크기를 기준으로 요금이 계산됨). 자세한 내용은 가격 | Cloud Object Storage를 참고하십시오.

MAZ_INTELLIGENT TIERING/INTELLIGENT TIERING

MAZ_INTELLIGENT TIERING 스토리지 클래스의 객체는 MAZ_STANDARD 및 MAZ_STANDARD_IA의 두 스토리지 레이어에 저장할 수 있습니다. INTELLIGENT_TIERING 스토리지 클래스의 객체는 STANDARD 및 STANDARD_IA의 두 스토리지 클래스에도 저장할 수 있습니다. COS는 데이터 검색 요금 발생 없이 객체의 액세스 빈도에 따라 스토리지 클래스 간에 자동으로 전환하므로 스토리지 요금이 절감됩니다. 자세한 내용은 INTELLIGENT TIERING 개요를 참고하십시오.

MAZ_INTELLIGENT TIERING은 INTELLIGENT TIERING과 다른 스토리지 메커니즘을 사용하여 동일한 리전의 여러 데이터 센터에 데이터를 저장하여 한 데이터 센터의 장애가 전체 서비스에 영향을 미치지 않도록 방지하고 비즈니스 안정성을 더욱 보장합니다.

적용 시나리오

데이터 액세스 패턴이 불확실한 시나리오에 적합합니다. 비즈니스에서 비용을 엄격하게 제어하고 파일 읽기 성능에 덜 민감한 경우 MAZ_INTELLIGENT TIERING 또는 INTELLIGENT TIERING을 사용하여 비용을 줄일 수 있습니다.

> 주의 :

MAZ_INTELLIGENT TIERING 및 INTELLIGENT_TIERING의 경우 객체는 실제 크기를 기준으로 요금이 과금됩니다. 가격 책정에 대한 자세한 내용은 가격 | Cloud Object Storage를 참고하십시오.

ARCHIVE

COS ARCHIVE는 콜드 데이터용으로 설계된 매우 안정적인 객체 스토리지 서비스로 스토리지 비용이 매우 낮고 데이터 장기 보존이 가능합니다. 이 스토리지 클래스의 최소 스토리지 기간은 90일입니다. ARCHIVE에 저장된 데이터를 읽기 위해서는 먼저 STANDARD로 복원해야 합니다.

COS는 ARCHIVE에 대해 다음 세 가지 복구 모드를 지원합니다.

- 고속 검색 모드: 1-5분 이내에 객체를 복구합니다.
- 일반 검색 모드: 3 - 5시간 이내에 객체를 복구합니다.
- 대량 검색 모드: 5 - 12시간 이내에 여러 객체를 복구합니다.

설명：

- 객체 복구에 대한 자세한 내용은 아카이브 객체 복구를 참고하십시오.
- 데이터 복구 요청에는 QPS 제한이 있으며, 100회/초로 제한됩니다.

적용 시나리오

보관 데이터, 의료 영상, 과학 데이터 등 컴플라이언스 파일 보관, 라이프사이클 파일 보관, 작업 로그 보관, 원격 재해 복구 등 데이터를 장기간 저장해야 하는 시나리오에 적합합니다.

주의：
ARCHIVE에는 최소 스토리지 요구 사항이 있습니다. 보관 기간이 90일 미만인 경우 청구서는 90일로 계산됩니다. 마찬가지로 파일 크기가 64KB 미만인 경우 요금은 64KB로 계산됩니다(파일 크기가 64KB보다 크거나 같으면 실제 파일 크기를 기준으로 요금이 계산됨). 자세한 내용은 가격 | Cloud Object Storage를 참고하십시오.

DEEP ARCHIVE

COS DEEP ARCHIVE는 가장 낮은 스토리지 비용과 장기 데이터 보관을 제공하는 매우 안정적인 객체 스토리지 서비스입니다. 이 스토리지 클래스의 최소 보관 기간은 180일입니다. DEEP ARCHIVE에 저장된 데이터를 읽으려면 먼저

STANDARD로 복구해야 합니다. 자세한 내용은 DEEP ARCHIVE 소개를 참고하십시오.

COS는 DEEP ARCHIVE에 대해 다음 두 가지 복구 모드를 지원합니다.

- 일반 검색 모드: 12 - 24시간 이내에 객체를 복구합니다.
- 대량 검색 모드: 24 - 48시간 이내에 여러 객체를 복구합니다.

> 설명 :
> 데이터 복구 요청에는 QPS 제한이 있으며, 100회/초로 제한됩니다.

적용 시나리오

의료 영상, 뷰 데이터, 로그 데이터와 같은 장기 저장이 필요한 비즈니스 시나리오에 사용됩니다.

> 주의 :
> DEEP ARCHIVE는 최소 스토리지 요구 사항이 있습니다. 보관 기간이 180일 미만인 경우 청구서는 180일로 계산됩니다. 마찬가지로 파일 크기가 64KB 미만인 경우 요금은 64KB로 계산됩니다(파일 크기가 64KB보다 크거나 같으면 실제 파일 크기를 기준으로 요금이 계산됨). 자세한 내용은 가격 | Cloud Object Storage를 참고하십시오.

스토리지 유형 비교

비교 항목	MAZ_STANDARD	MAZ_STANDARD_IA	MAZ_INTELLIGENT TIERING	INTELLIGENT TIER
스토리지 클래스 매개변수	MAZ_STANDARD	MAZ_STANDARD_IA	MAZ_INTELLIGENT_TIERING	INTELLIGENT_TIE
데이터 내구성	99.9999999 999%	99.9999999 999%	99.9999999 999%	99.9999999 99%

비교 항목	MAZ_STANDARD	MAZ_STANDARD_IA	MAZ_INTELLIGENT TIERING	INTELLIGENT TIER
서비스 가용성	99.995%	99.995%	99.995%	99.99%
응답	밀리초	밀리초	밀리초	밀리초
과금 가능한 최소 객체 크기	실제 객체 크기로 계산	64KB	실제 객체 크기로 계산	실제 객체 크기로 계
최소 보관 기간	제한 없음	30일	제한 없음	제한 없음

비교 항목	MAZ_STANDARD	MAZ_STANDARD_IA	MAZ_INTELLIGENT TIERING	INTELLIGENT TIER
지원 리전	베이징, 상하이, 광저우, 중국홍콩, 싱가포르만 해당	베이징, 상하이, 광저우, 중국홍콩, 싱가포르만 해당	베이징, 상하이, 광저우, 싱가포르만 해당	베이징, 난징, 상하이 저우, 청두, 충칭, 도 싱가포르만 해당
스토리지 요금	비교적 높음	비교적 높음	인텔리전트 티어링 후 스토리지 클래스에 따라 다름	인텔리전트 티어링 토리지 클래스에 따 름
데이터 검색 요금	없음	비교적 낮음, 실제 읽은 데이터 양에 따라 과금	없음	없음
요청 요금	일반	비교적 높음	비교적 높음, 또한 INTELLIGENT TIERING 객체 모니터링 요금이 과금됨	비교적 높음, 또한 INTELLIGENT TIER 객체 모니터링 요금 금됨
데이터 처리	지원	지원	지원	지원

스토리지 유형의 전환

COS는 활성 객체가 COS에 있는 방식을 나타내는 다양한 스토리지 유형을 제공합니다. 여전히 필요에 따라 객체의 스토리지 유형을 변경하거나 객체를 STANDARD_IA, ARCHIVE 또는 DEEP ARCHIVE와 같은 덜 활성화된 스토리지 유형으로 전환할 수 있습니다.

설명：

- 객체를 전환할 때 객체가 상주하는 리전에 대해 대상 스토리지 유형이 지원되는지 확인하십시오.
- **ARCHIVE/DEEP ARCHIVE**에 저장된 객체의 스토리지 유형을 수정하려면 먼저 STANDARD로 복구해야 합니다. 자세한 내용은 아카이브된 객체 복구를 참고하십시오.

각 스토리지 유형을 전환하는 방법은 다음과 같습니다.

스토리지 유형	변경 가능한 스토리지 유형	전환 가능한 스토리지 유형과 전환 우선순위
MAZ_STANDARD	MAZ_STANDARD_IA, MAZ_INTELLIGENT TIERING	MAZ_STANDARD > MAZ_STANDARD_IA > MAZ_INTELLIGENT TIERING
MAZ_STANDARD_IA	MAZ_STANDARD, MAZ_INTELLIGENT TIERING	MAZ_STANDARD_IA > MAZ_INTELLIGENT TIERING
MAZ_INTELLIGENT TIERING	MAZ_STANDARD, MAZ_STANDARD_IA	없음
INTELLIGENT TIERING	STANDARD, STANDARD_IA, ARCHIVE, DEEP ARCHIVE	INTELLIGENT TIERING > ARCHIVE > DEEP ARCHIVE
STANDARD	INTELLIGENT TIERING, STANDARD_IA, ARCHIVE, DEEP ARCHIVE	STANDARD > STANDARD_IA > INTELLIGENT TIERING > ARCHIVE > DEEP ARCHIVE
STANDARD_IA	INTELLIGENT TIERING, STANDARD, ARCHIVE, DEEP ARCHIVE	STANDARD_IA > INTELLIGENT TIERING > ARCHIVE > DEEP ARCHIVE
ARCHIVE	INTELLIGENT TIERING, STANDARD, STANDARD_IA, DEEP ARCHIVE (복구 후)	ARCHIVE > DEEP ARCHIVE
DEEP ARCHIVE	INTELLIGENT TIERING, STANDARD, STANDARD_IA, ARCHIVE (복구 후)	없음

구체적인 작업 가이드는 다음 문서를 참고하십시오.

- 스토리지 유형 수정
- 라이프사이클 설정

DEEP ARCHIVE 소개

최종 업데이트 날짜: : 2023-03-14 14:44:27

소개

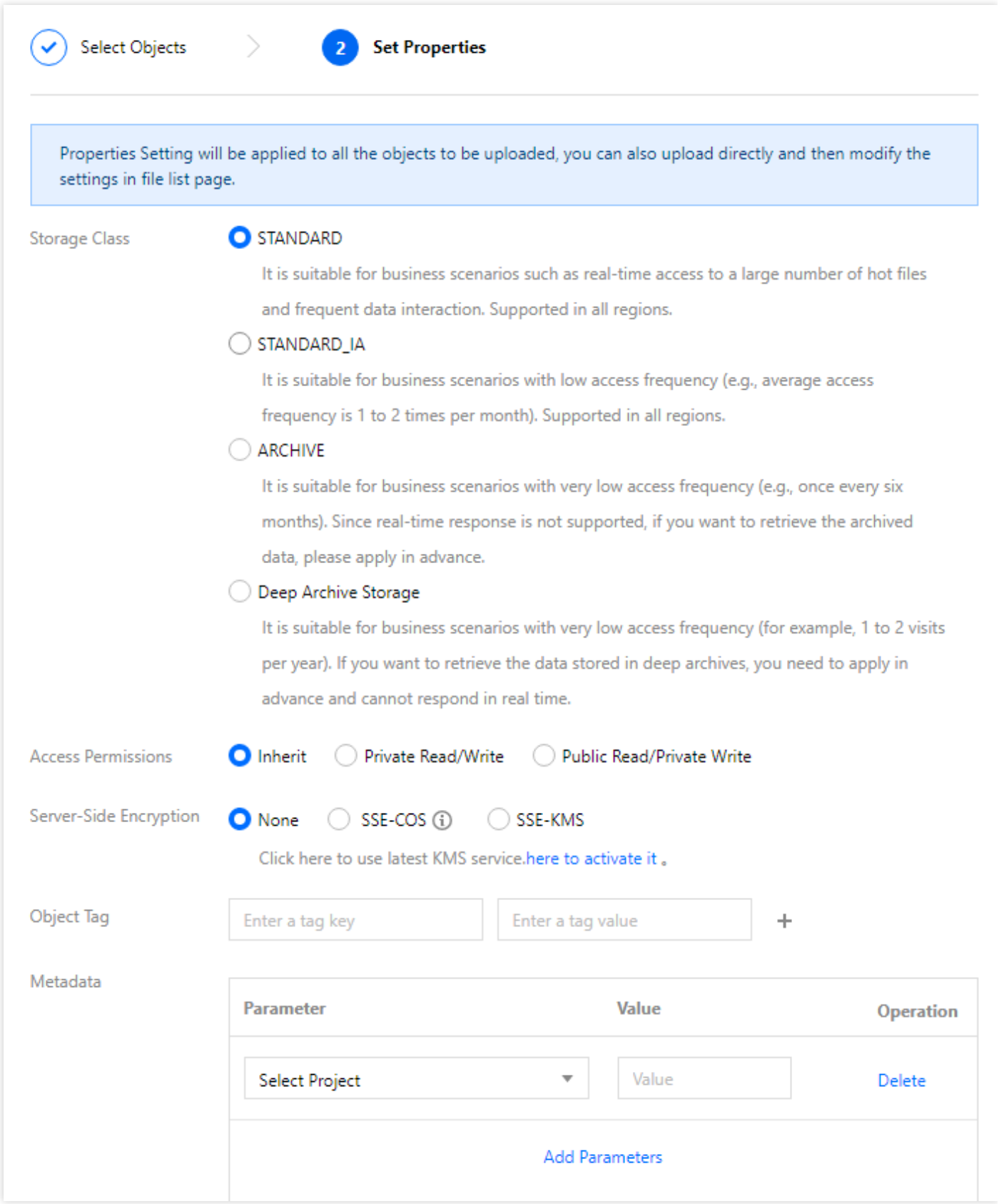
DEEP ARCHIVE는 Cloud Object Storage(COS)가 제공하는 보관 서비스로, 방대한 데이터를 장기간 보관할 수 있는 서비스입니다. DEEP ARCHIVE는 테이프 스토리지 등급별 스토리지 단가를 제공하며, 사용자의 데이터 장기 저장을 위한 저비용 솔루션을 제공합니다. 사용자는 로컬에서 복잡한 테이프 라이브러리 설정을 점검하거나 기저에 있는 스토리지 미디어의 발전에 관심을 기울일 필요가 없습니다. 객체 스토리지 COS가 제공하는 API, SDK, 생태 툴 및 콘솔과 같이 사용자와 컴퓨터 간의 풍부한 인터랙티브 수단을 통해 데이터를 저비용으로 간편하게 관리할 수 있기 때문입니다.

DEEP ARCHIVE는 데이터 액세스 빈도는 매우 낮지만 장기 보관이 필요한 시나리오에 적합합니다. 로그 콜드 스탠바이 시나리오에서 기업은 현지 법규에 따라 매일 발생하는 로그 데이터에 콜드 스탠바이를 실행하여 쉽게 추적 및 분석할 수 있습니다. 데이터 뷰 및 자율 주행 등의 비즈니스에서 기업은 대량의 이미지, 비디오 등 미디어 파일을 누적하게 되며 데이터를 사용한 후에도 장기간 보관하고 저장해야 합니다. 기업은 DEEP ARCHIVE를 통해 이러한 데이터를 클라우드에 저장하고 필요할 때만 복구하여 스토리지 비용을 낮추고 유지보수 관리의 어려움을 줄일 수 있습니다.

DEEP ARCHIVE는 99.999999999%의 내구성과 99.95%의 가용성을 위해 설계되었습니다. 또한 COS에서 제공하는 버전 관리 및 리전 간 복제 기능을 사용하여 데이터 보안을 추가로 보장할 수 있습니다.

사용 방법

1. 콘솔을 사용한 업로드
2. COS 콘솔에 로그인합니다.
3. **버킷 리스트**를 클릭하여 지원되는 리전(예시: 베이징 리전)에 버킷을 생성합니다.
4. 생성이 완료되면 **파일 리스트** 인터페이스에서 **파일 업로드**를 클릭합니다.
5. 업로드할 로컬 파일을 선택하고, 객체 속성 설정에서 스토리지 유형을 **DEEP ARCHIVE**로 선택합니다. 자세한 내용은 객체 업로드를 참고하십시오.



6. API를 사용한 업로드

PUT Object, POST Object 또는 Initiate Multipart Upload 인터페이스에서 `x-cos-storage-class` 를

`DEEP_ARCHIVE` 로 설정하여 다이렉트 DEEP ARCHIVE를 구현합니다.

> 주의：
> Append Object는 DEEP ARCHIVE 유형의 직접 전송을 지원하지 않습니다.

7. SDK를 사용한 업로드
   현재 COS에서 출시한 모든 SDK는 DEEP ARCHIVE로의 직접 업로드를 지원합니다. 파일을 업로드할 때 `StorageClass` 매개변수를 `DEEP_ARCHIVE` 로 설정해야 합니다.
8. 툴을 사용한 업로드
   COSBrowser, COSCMD 툴은 다이렉트 DEEP ARCHIVE를 지원합니다. 구체적인 방법은 파일 업로드 시, 헤더 필드 `x-cos-storage-class` 를 늘리고 `DEEP_ARCHIVE` 로 설정해 다이렉트 DEEP ARCHIVE를 구현하는 것입니다.

사용 제한

DEEP ARCHIVE는 다음과 같은 제한이 있습니다. 사용 과정에서 관련 제한이 스토리지 비용과 성능에 미치는 영향에 유의하십시오.

- **최소 스토리지 단위 제한**: 최소 스토리지 단위는 64KB로, 64KB 이하의 객체는 64KB로 스토리지 용량을 계산합니다.
- **최단 스토리지 시간 제한**: 최단 스토리지 시간은 180일로, 180일 이하는 180일로 과금됩니다.

> 설명：
> 스토리지 일수는 로직에 따라 24시간을 1일로 하며, 카운트 시작 시간은 파일 수정 시간으로 계산합니다.

- **검색 요청 QPS 제한**: 100 요청/초.
- **지원 리전**: 현재 베이징, 난징, 상하이, 광저우, 청두, 충칭, 도쿄 및 싱가포르 리전만 지원되며, 기타 리전은 순차적으로 지원 예정입니다.
- **사용 제한**: 현재 DEEP ARCHIVE는 다중 AZ 버킷에 대해 지원되지 않습니다.
- **작업 제한**: DEEP ARCHIVE 스토리지 클래스의 다른 객체에 객체를 추가할 수 없습니다.
- **검색 요청 제한**: 객체에 대해 한 번에 하나의 검색 요청만 실행할 수 있으며, 여러 반복 요청은 병합되어 가장 빠른 검색 모드에 따라 처리됩니다. (N+1)번째 요청의 검색 모드가 N번째 요청보다 빠르면 (N+1)번째 요청이 새 요청으로 실행됩니다. 그렇지 않으면 실패합니다.
- **데이터 검색**: DEEP ARCHIVE 스토리지 클래스의 파일에 대해 표준 검색(12 - 24시간) 및 대량 검색(24 - 48시간)이 지원됩니다.

INTELLIGENT TIERING 개요

최종 업데이트 날짜: : 2023-04-28 15:31:50

개요

INTELLIGENT TIERING 유형은 데이터에 콜드/핫 레이어 메커니즘을 구분해 사용자 데이터의 액세스 모드에 따라 자동으로 데이터의 콜드/핫 레이어를 전환함으로써 사용자 데이터의 스토리지 비용을 줄일 수 있습니다.

INTELLIGENT TIERING은 알 수 없거나 액세스 패턴이 변경되는 데이터에 이상적입니다. STANDARD와 동일한 짧은 대기 시간과 높은 처리량을 제공하는 동시에 비용이 저렴합니다. 클라우드 스토리지 비용을 줄이기 위해 필요에 따라 액세스 패턴이 불확실한 객체의 스토리지 클래스를 STANDARD에서 INTELLIGENT TIERING으로 변경할 수 있습니다.

주의 :

- INTELLIGENT TIERING은 현재 베이징, 난징, 상하이, 광저우, 청두, 충칭, 도쿄 및 싱가포르 리전에서 지원됩니다. MAZ_INTELLIGENT TIERING은 베이징, 상하이, 광저우 및 싱가포르 리전에서만 사용할 수 있습니다.
- INTELLIGENT TIERING은 INTELLIGENT TIERING에 대한 스토리지 용량 및 객체 모니터링에 대한 요금이 발생하는 독립형 스토리지 클래스입니다. INTELLIGENT TIERING 스토리지 팩을 구매하여 사용할 수 있습니다. 자세한 가격은 제품 가격을 참고하십시오.

강점

사용자가 데이터를 업로드할 때 INTELLIGENT TIERING 유형을 선택해 COS에 보관할 경우, COS는 주기적으로 데이터 액세스 횟수를 모니터링하며 데이터 액세스가 일정 기간 지속되지 않을 경우 데이터를 스토리지 비용이 더 저렴한 액세스 레이어로 이동합니다. 데이터가 다시 액세스되면, 다시 고빈도 액세스 레이어로 이동해 데이터 읽기 성능을 보장합니다. 데이터 핫/콜드 티어링 스토리지를 통해 사용자가 스토리지 비용과 읽기 성능 사이에서 균형점을 찾을 수 있게 도와줍니다. INTELLIGENT TIERING은 다음과 같은 장점이 있습니다.

- **비용 집약**: 데이터를 지속적으로 INTELLIGENT TIERING 유형으로 저장할 경우, 저장 시간이 길어질수록 STANDARD 스토리지 비용은 상대적으로 낮아져 최대 20% 정도의 스토리지 비용을 절약할 수 있습니다. INTELLIGENT TIERING 유형은 객체 스토리지 라이프사이클 프로세스에 관여하여, 사용자는 필요에 따라 INTELLIGENT TIERING을 ARCHIVE로 전환해 데이터 저장 비용을 더 낮출 수 있습니다.
- **안정성 및 내구성**: INTELLIGENT TIERING은 STANDARD 스토리지와 동일한 짧은 딜레이 시간, 높은 처리량을 제공합니다. 또한 삭제 코드를 사용하여 중복성을 달성함으로써 최대 99.999999999%(11개 9)의 안정성과 블록

스토리지 및 동시 읽기/쓰기를 사용하여 최대 99.95%의 가용성을 제공합니다. MAZ_INTELLIGENT TIERING은 최대 99.9999999999%(12개 9)의 안정성과 최대 99.995%의 가용성을 제공합니다.

- **편리성**: 데이터에 객체 스토리지 유형만 지정하면 INTELLIGENT TIERING의 특징을 적용할 수 있습니다. INTELLIGENT TIERING 스토리지 유형은 COS의 API, SDK, 툴 및 생태 애플리케이션에 자연스럽게 융합되어, 사용자가 필요에 따라 클라우드에 저장된 데이터를 관리할 수 있게 합니다.

사용 방법

데이터를 INTELLIGENT TIERING 유형으로 COS에 저장하려면 먼저 버킷의 INTELLIGENT TIERING 설정을 활성화해야 합니다. 활성화 후, 사용자가 객체를 업로드할 때 **스토리지 유형**을 INTELLIGENT TIERING 유형으로 지정하면 됩니다.

COS 콘솔 사용

객체 업로드 시 INTELLIGENT TIERING으로 설정

다음 단계를 참고하여 객체를 INTELLIGENT TIERING 유형으로 저장할 수 있습니다.

1. 버킷 설정 페이지에서 INTELLIGENT TIERING 설정을 활성화합니다. 자세한 내용은 INTELLIGENT TIERING 설정 문서를 참고하십시오.
2. 파일을 업로드하고 업로드 중에 스토리지 클래스를 지정합니다. 파일 업로드 방법에 대한 자세한 내용은 객체 업로드를 참고하십시오.

> 주의 :
> 버킷의 INTELLIGENT TIERING 구성을 활성화하면 비활성화할 수 없으니 신중하게 설정하십시오.

클라우드 상의 데이터를 INTELLIGENT TIERING으로 전환

다음 단계를 참고하여 업로드된 인벤토리 데이터를 INTELLIGENT TIERING 유형으로 전환할 수 있습니다.

1. 버킷 설정 페이지에서 라이프사이클 규칙을 생성합니다. 자세한 내용은 라이프사이클 설정 문서를 참고하십시오.
2. 지정된 규칙의 응용 범위를 설정하고 데이터를 INTELLIGENT TIERING으로 전환합니다.

REST API 사용

다음 API를 통해 INTELLIGENT TIERING을 직접 설정할 수 있습니다.

1. 먼저 REST API를 사용해 버킷의 INTELLIGENT TIERING을 활성화합니다. 다음 API 문서를 참고하십시오.
   - PUT Bucket IntelligentTiering
   - GET Bucket IntelligentTiering

2. 버킷에 INTELLIGENT TIERING이 활성화되면, 다음 API 문서를 참고하여 객체를 INTELLIGENT TIERING 유형으로 업로드할 수 있습니다.
   - PUT Object
   - PUT Object - Copy
   - POST Object
   - Initiate Multipart Upload
3. 객체의 스토리지 유형 및 속한 스토리지 레이어를 조회해야 할 경우, 다음 API 문서를 참고하십시오.
   - GET Object
   - HEAD Object
4. REST API를 직접 사용하여 INTELLIGENT TIERING 유형의 객체를 삭제할 수 있습니다. 다음 API 문서를 참고하십시오.
   - DELETE Object
   - DELETE Multiple Objects

SDK 사용

현재 모든 COS SDK는 INTELLIGENT TIERING 및 MAZ_INTELLIGENT TIERING을 지원합니다. 이러한 스토리지 클래스를 사용하려면 파일을 업로드할 때 StorageClass를 INTELLIGENT_TIERING 또는 MAZ_INTELLIGENT_TIERING으로 설정하십시오. 객체 업로드 SDK 문서는 객체 업로드를 참고하십시오.

사용 제한

INTELLIGENT TIERING 사용에는 다음과 같은 제한이 있습니다.

- **구성 제한**: 구성된 후에는 수정할 수 없습니다. 수정하려면 문의하기를 통해 연락주십시오. 저빈도 액세스 레이어 전환 일수 선택 값은 30, 60, 90입니다.
- **초기 스토리지 레이어 제한**: INTELLIGENT TIERING 유형의 신규 객체는 기본적으로 고빈도 액세스 레이어에 저장됩니다. 일정 기간 동안 액세스가 없는 상태가 지속되어야만 저빈도 액세스 레이어로 전환됩니다.
- **최소 스토리지 단위 제한**: 64KB 이하 객체는 고빈도 액세스 레이어에 영구 저장되며, 고빈도 액세스 레이어와 저빈도 액세스 레이어 간 전환은 불가합니다. 단일 스토리지 파일의 크기에 관계없이 실제 데이터 크기에 따라 과금됩니다.
- **작업 제한**: 추가 업로드 인터페이스 통해 객체를 INTELLIGENT TIERING 유형으로 업로드하는 기능은 지원하지 않습니다.
- **라이프사이클 제한**: INTELLIGENT TIERING 유형은 ARCHIVE 또는 DEEP ARCHIVE 유형으로만 전환될 수 있습니다. STANDARD 스토리지 유형이 INTELLIGENT TIERING 유형으로 전환될 때는 고빈도 액세스 레이어에 저장되며, STANDARD_IA 스토리지 유형이 INTELLIGENT TIERING 유형으로 전환될 때는 STANDARD_IA 스토리지 액세스 레이어에 저장됩니다.

- **버킷 복사 제한**: 버킷 복사 시, 타깃 버킷의 INTELLIGENT TIERING 설정이 활성화되지 않았을 경우 객체를 INTELLIGENT TIERING 유형으로 복사할 수 없습니다.

FAQ

INTELLIGENT TIERING은 어떻게 과금되나요?

INTELLIGENT TIERING에는 **INTELLIGENT TIERING 용량 요금** 및 **INTELLIGENT TIERING 객체 모니터링 요금**이 포함됩니다. 그 중:

1. INTELLIGENT TIERING 용량 요금은 파일이 위치한 스토리지 레이어에 따라 다른 스토리지 요금을 부과합니다.

- 파일이 고빈도 레이어에 있을 경우 STANDARD 스토리지 요금에 따라 과금됩니다.
- 파일이 STANDARD_IA 레이어에 있을 경우 STANDARD_IA 스토리지 용량 요금에 따라 과금됩니다.

> 설명 :
>
> - STANDARD 스토리지 및 STANDARD_IA 스토리지 용량 요금은 퍼블릭 클라우드 리전에 따라 가격이 상이하며, 구체적인 가격은 가격 | Cloud Object Storage를 참고하십시오.
> - 파일 업로드 및 다운로드 과정에서 요청 요금 및 트래픽 요금도 발생하며, 이러한 요금 계산 예시는 트래픽 요금 과금 사례 및 요청 요금 과금 사례를 참고하십시오.

2. INTELLIGENT TIERING 객체 모니터링 요금은 저장된 객체 수에 따라 부과됩니다(64KB 미만 파일 제외). 자세한 가격은 가격 | Cloud Object Storage를 참고하십시오.

예시

한 회사에 10만 개의 객체(모두 64KB 이상, 총 1TB)가 있고 데이터가 베이징 리전의 INTELLIGENT TIERING 스토리지 클래스에 저장되고 30일 후에 자주 액세스하지 않는 티어로 전환된다고 가정합니다. 객체의 20%(즉, 객체 2만 개)가 30일마다 비정기 액세스 티어로 전환되는 경우 **매 30일 간의** 객체 모니터링 요금 및 스토리지 사용 요금은 다음과 같습니다.

> 설명 :
>
> 아래 표에서 베이징 리전 객체 모니터링 요금의 월 단가는 0.25 USD/만 개 객체이며, '일 단가 = 월 단가 / 30'의 변환 로직에 따르면 일 단가는 0.00833333 USD/만 개 객체/일입니다.

저장 일수	객체 모니터링 요금(USD)/30일	INTELLIGENT TIERING 스토리지 사용 요금(USD)/30일	STANDARD 스토리지 사용 요금(USD)/30일
30 x 1	0.25 USD/만 개 객체 x 10만	1024 x 0.024 / 30 x 30 = 24.58	1024 x 0.024 / 30 x 30 = 24.58
30 x 2	0.25 USD/만 개 객체 x 10만	819.2 x 0.024 / 30 x 30 + 204.8 x 0.08 / 30 x 30 = 23.35	1024 x 0.024 / 30 x 30 = 24.58
30 x 3	0.25 USD/만 개 객체 x 10만	655.36 x 0.024 / 30 x 30 + 368.64 x 0.08 / 30 x 30 = 22.36	1024 x 0.024 / 30 x 30 = 24.58
30 x 4	0.25 USD/만 개 객체 x 10만	524.288 x 0.024 / 30 x 30 + 499.712 x 0.08 / 30 x 30 = 21.58	1024 x 0.024 / 30 x 30 = 24.58
30 x 5	0.25 USD/만 개 객체 x 10만	419.4304 x 0.024 / 30 x 30 + 604.5696 x 0.08 / 30 x 30 = 20.95	1024 x 0.024 / 30 x 30 = 24.58
30 x 6	0.25 USD/만 개 객체 x 10만	335.54432 x 0.024 / 30 x 30 + 688.45568 x 0.08 / 30 x 30 = 20.45	1024 x 0.024 / 30 x 30 = 24.58

스토리지 기간이 길어질수록 30일 마다 소액의 모니터링 비용만 지불하면 되므로 상당한 비용 절감 효과를 얻을 수 있음을 알 수 있습니다.

INTELLIGENT TIERING은 어떤 유형의 파일에 적용됩니까?

INTELLIGENT TIERING은 오디오/비디오, 로그와 같이 파일 크기가 평균적으로 큰 파일에 적합하며 액세스 모드는 고정되어 있지 않습니다. 평균 파일 용량이 클수록 각 파일의 GB당 지불해야 하는 모니터링 비용이 줄어듭니다. 비즈니스 액세스 모드가 상대적으로 고정되어 있으면 INTELLIGENT TIERING을 사용할 필요 없이 라이프사이클을 통해 지정된 시간을 설정하여 STANDARD_IA 스토리지로 전환할 수 있습니다.

INTELLIGENT TIERING으로 파일을 저장하는 방법은 무엇입니까?

다음 두 가지 방법으로 파일을 INTELLIGENT TIERING으로 저장할 수 있습니다.

- 추가 파일: 업로드 시 스토리지 유형을 INTELLIGENT TIERING으로 지정하기만 하면 파일을 INTELLIGENT TIERING으로 저장할 수 있습니다.
- 기존 파일: COPY 인터페이스를 통해 파일 스토리지 유형을 INTELLIGENT TIERING 유형으로 수정하거나 라이프사이클 기능을 사용하여 STANDARD 스토리지 및 STANDARD_IA 스토리지 유형을 INTELLIGENT TIERING 유형으로 전환할 수 있습니다.

주의 :

64KB보다 작은 INTELLIGENT TIERING 객체는 항상 STANDARD에 저장됩니다. 이러한 객체의 경우 비용 절감을 위해 필요에 따라 STANDARD, STANDARD_IA, ARCHIVE 또는 DEEP ARCHIVE 스토리지 클래스에 업로드하는 것이 좋습니다.

INTELLIGENT TIERING 설정을 끄는 방법은 무엇입니까?

INTELLIGENT TIERING은 활성화 후 **비활성화할 수 없습니다**. 파일을 INTELLIGENT TIERING으로 저장할 필요가 없는 경우 파일을 업로드할 때 파일 스토리지 유형을 STANDARD 스토리지, STANDARD_IA 스토리지, ARCHIVE 또는 DEEP ARCHIVE로 지정하기만 하면 됩니다.

객체 업로드
간편 업로드

최종 업데이트 날짜: : 2022-12-02 14:21:15

간편 업로드란 PUT Object 인터페이스를 사용해 객체를 업로드하는 것을 의미하며, 5GB 이하 단일 객체 업로드 요청에 사용됩니다.

5GB를 초과하는 단일 객체는 다음 방법을 사용하여 업로드할 수 있습니다.

- 콘솔을 통한 업로드: 콘솔을 이용해 최대 512GB의 단일 객체를 업로드할 수 있습니다. 자세한 내용은 객체 업로드 콘솔 가이드 문서를 참조하십시오.
- API/SDK를 통한 멀티파트 업로드: 최대 48.82TB(50000GB)의 단일 객체를 업로드할 수 있습니다. 자세한 내용은 멀티파트 업로드를 참조하십시오.
- COSCMD 툴을 사용해 최대 40TB의 단일 객체를 업로드할 수 있습니다. 자세한 내용은 COSCMD 툴을 참조하십시오.

> 설명 :
> 지정된 폴더 또는 경로로 업로드를 요청하는 경우 `/` 를 사용할 수 있습니다(예시: picture.png를 doc 폴더에 업로드 하는 경우 객체 키를 doc/picture.png로 설정).

적용 시나리오

간편 업로드는 5GB 이하의 객체 업로드에 활용할 수 있습니다.

고대역폭 혹은 약한 네트워크 환경에서 업로드하는 객체 크기가 큰 경우(100MB 이상~5GB 이하인 경우) 우선적으로 멀티파트 업로드 사용을 권장합니다. 멀티파트 업로드는 여러 개의 파트가 병렬 전송되며, 고대역폭 환경에서 리소스를 효과적으로 이용할 수 있습니다. 약한 네트워크 환경에서 단일 파트 업로드에 실패한 경우, 이미 업로드된 다른 파트에 영향을 주지 않기 때문에 실패한 부분만 다시 업로드하여 전체 업로드 성공률을 높여 줍니다. 약한 네트워크 환경에서의 모바일 업로드에 대한 자세한 내용은 약한 네트워크에서의 멀티파트 업로드 재개 실행을 참조하십시오.

사용 방법

REST API 사용

REST API를 통해 직접 객체 간편 업로드 요청을 보낼 수 있습니다. 자세한 내용은 PUT Object API 문서를 참조하십시오.

SDK 사용

SDK 호출을 통해 직접 객체 간편 업로드를 진행할 수 있습니다. 자세한 내용은 다음 언어별 SDK 문서를 참조하십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- Mini Program SDK

멀티파트 업로드

최종 업데이트 날짜: : 2022-12-02 14:17:00

소개

멀티파트 업로드는 전체 객체를 여러 파트로 나누어 Cloud Object Storage(COS)에 멀티파트 업로드할 수 있습니다. 업로드 시 연속 일련번호에 따라 단일 업로드하거나 임의 순서대로 파트를 각각 업로드하면 마지막에 COS에서 파트 일련번호 순서대로 객체를 다시 결합합니다. 전송 실패한 파트가 생기더라도 재전송할 수 있으며 다른 파트나 콘텐츠 완성도에 영향을 주지 않습니다. 일반적으로 약한 네트워크 환경에서 단일 객체가 20MB보다 크면 우선적으로 멀티파트 업로드를 고려할 수 있으며 고대역폭에서는 100MB가 넘는 객체는 멀티파트 업로드를 진행합니다.

멀티파트 업로드는 큰 객체를 최대 10000개 파트로 분할할 수 있습니다. 분할한 파트의 크기는 일반적으로 1MB - 5GB입니다. 마지막 파트는 1MB보다 작을 수 있습니다.

> 설명 :
> 간단 업로드는 최대 5GB 파일까지 업로드할 수 있지만 멀티파트 업로드는 5GB 이상파일도 업로드 가능합니다.

적용 시나리오

멀티파트 업로드는 약한 네트워크 혹은 고대역폭 환경에서 크기가 비교적 큰 객체를 업로드할 경우 활용할 수 있습니다.

멀티파트 업로드의 장점은 다음과 같습니다.

- 약한 네트워크 환경에서 작은 크기의 파트로로 네트워크 연결 실패로 인한 업로드 중단 현상을 줄이고 객체 전송을 지속할 수 있습니다.
- 고대역폭 환경에서 객체 파트 동시 업로드 시 네트워크 대역폭을 충분히 이용할 수 있으며 비순차적 업로드가 최종 결합한 객체에 영향을 주지 않습니다.
- 멀티파트 업로드를 사용하면 언제든지 업로드를 중지하고 단일 대형 객체 업로드를 재개할 수 있습니다. 작업을 끝내지 않는 이상 작업 완료되지 않은 모든 객체는 언제든지 업로드를 지속할 수 있습니다.
- 멀티파트 업로드는 객체의 전체 크기를 모르는 상황에서 업로드할 때 활용하기도 합니다. 먼저 멀티파트 업로드를 진행한 후 객체를 결합하여 전체 크기를 얻을 수 있습니다.

사용 방법

REST API 사용

REST API를 통해 즉각적으로 멀티파트 업로드 요청을 보낼 수 있습니다. 자세한 내용은 다음 API 문서를 참조하십시오.

- Initiate Multipart Upload
- Upload Part
- Complete Multipart Upload
- Abort Multipart Upload

SDK 사용

SDK 호출을 통해 즉각적으로 멀티파트 작업을 진행할 수 있습니다. 자세한 내용은 다음 언어별 SDK 문서를 참조하십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- Mini Program SDK

사전 서명된 URL을 통해 업로드

최종 업데이트 날짜: : 2022-01-27 12:34:47

적용 시나리오

기본적으로 버킷과 객체는 개인 소유입니다. 서드 파티가 버킷에 객체를 업로드할 수 있지만 CAM 계정 혹은 임시 키 사용을 원치 않는 경우 임시 업로드 작업을 완료하기 위해 URL 사전 서명을 사용해 서드 파티에게 서명을 제출합니다. 유효한 서명 URL을 받은 계정은 모두 객체 업로드를 진행할 수 있습니다.

URL 사전 서명 시 서명에 객체 키를 설정하여 지정한 객체 키만 업로드할 수 있도록 허용합니다. 절차에서 사전 서명 URL의 유효 기간을 지정하여 해당 URL 기간 만료 후 사용 권한이 없는 계정의 사용을 막을 수 있습니다.

설명 :

- 사용자는 임시 키를 사용하여 사전 서명을 생성하고, 임시 승인을 통해 사전 서명 업로드 및 다운로드 요청의 보안성을 강화할 것을 권장합니다. 임시 키 신청 시, 최소 권한의 원칙 관련 가이드를 준수하여 타깃 버킷이나 객체 이외의 리소스가 유출되지 않도록 하시기 바랍니다.
- 사전 서명 생성을 위해 영구 키를 사용해야 하는 경우, 리스크 방지를 위해 영구 키 권한을 업로드 또는 다운로드 작업으로 제한할 것을 권장합니다.

사용 방법

SDK 사용

SDK에서 사전 서명된 URL 메소드를 직접 호출할 수 있습니다. 자세한 내용은 다음 언어별 SDK 문서를 참고하십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK

- PHP SDK
- Python SDK
- 미니프로그램 SDK

객체 다운로드
객체 간편 다운로드

최종 업데이트 날짜: : 2023-03-14 14:44:27

적용 시나리오

COS(Cloud Object Storage)에서 객체 다운로드 요청을 직접 시작할 수 있습니다. 다음 기능이 지원됩니다.

- 전체 객체 다운로드: GET 요청을 시작하여 전체 객체 데이터를 다운로드합니다.
- 객체의 일부 다운로드: GET 요청에서 Range 요청 헤더를 사용하여 객체의 특정 바이트 범위를 검색합니다. 여러 범위 검색은 지원되지 않습니다.

객체의 메타데이터는 객체의 콘텐츠와 함께 HTTP 응답 헤더로 반환됩니다. GET 요청은 URL 매개변수를 사용하여 응답에서 특정 메타데이터 값 덮어쓰기를 지원합니다.
예를 들어 Content-Disposition의 응답 값을 덮어쓸 수 있습니다. 수정을 지원하는 응답 헤더는 다음과 같습니다.

- Content-Type
- Content-Language
- Expires
- Cache-Control
- Content - Disposition
- Content-Encoding

사용 방법

COS 콘솔 사용

COS 콘솔에서 객체를 다운로드합니다. 자세한 내용은 콘솔 가이드의 객체 다운로드를 참고하십시오.

REST API 사용

REST API를 사용하여 객체 다운로드 요청을 시작합니다. 자세한 내용은 GET Object를 참고하십시오.

SDK 사용

SDK에서 객체 다운로드 메소드를 직접 호출합니다. 자세한 내용은 아래의 해당 프로그래밍 언어에 대한 SDK 문서를 참고하십시오.

- Android SDK

- C SDK
- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- 미니프로그램 SDK

사전 서명된 URL을 통해 다운로드

최종 업데이트 날짜: : 2023-04-28 15:30:49

사용 사례

버킷과 객체는 기본적으로 개인 소유입니다. 3rd party가 객체를 다운로드할 수 있도록 허용하지만 CAM 계정 또는 임시 키 사용을 제한하고 싶은 경우, URL 사전 서명 방식으로 3rd party에 서명을 제출함으로써 다운로드 작업을 완료할 수 있습니다. 유효한 URL 사전 서명을 받은 계정은 모두 객체 다운로드를 진행할 수 있습니다.

URL 사전 서명 시 서명에 객체 키를 설정하여 지정한 객체만 다운로드할 수 있도록 허용합니다. 또한 프로세스에서 URL 사전 서명 유효 기간을 지정하여, 해당 기간 만료 후 권한이 없는 계정의 사용을 제한할 수 있습니다.

주의 :

- CDN 도메인 이름에 액세스하려면 CDN 인증 프로세스를 따라야 하며, COS 서명을 사용할 수 없으므로 사전 서명된 URL은 CDN 도메인 이름을 사용할 수 없습니다.
- 사용자는 임시 키를 사용하여 사전 서명을 생성하고, 임시 승인을 통해 사전 서명 업로드 및 다운로드 요청의 보안성을 강화할 것을 권장합니다. 임시 키 신청 시, 최소 권한의 원칙 관련 가이드를 준수하여 타깃 버킷이나 객체 이외의 리소스가 유출되지 않도록 하시기 바랍니다.
- 사전 서명을 생성하기 위해 영구 키를 사용해야 하는 경우 위험을 방지하기 위해 영구 키의 권한 범위를 업로드 또는 다운로드 작업으로 제한하는 것이 좋습니다. 그리고 생성된 서명의 유효 기간은 업로드 또는 다운로드 작업을 완료하는 데 필요한 가장 짧은 기간으로 설정합니다. 지정된 사전 서명된 URL의 유효 기간이 만료되면 요청이 중단되기 때문에 새 서명을 신청한 후 실패한 요청은 다시 실행해야 하며, 체크포인트 재시작을 지원하지 않습니다.

사용 방법

SDK 사용

SDK의 URL 사전 서명 방식을 호출할 수 있습니다. 자세한 내용은 다음 언어별 SDK 문서를 참조하십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET SDK
- Flutter SDK

- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- React Native SDK
- 미니프로그램 SDK

객체 키 나열

최종 업데이트 날짜: : 2021-05-08 18:24:36

객체 키(ObjectKey)는 버킷에 있는 객체의 고유 식별자로, 일반적으로 파일 경로로 이해할 수 있습니다. 예를 들어 객체 키가 doc/picture.jpg라면, 이미지 파일 picture.jpg가 COS(Cloud Object Storage)의 doc 경로(또는 폴더)에 저장되어 있음을 의미합니다.

객체 키 나열은 지정한 객체 키에 따라 특정 객체를 불러오는 작업입니다. 이외에도 객체 키 접두사에 따라 객체를 불러올 수도 있습니다. 즉, 객체 키 앞 부분(예: doc)을 지정해 동일한 접두사인 doc를 가진 모든 객체를 불러올 수 있습니다.

적용 시나리오

Tencent Cloud COS는 접두사 순서에 따라 객체 키를 나열하므로, 객체 키에 `/` 부호를 사용하여 기존의 파일 시스템과 유사한 계층적 구조를 구현할 수 있습니다. COS도 세퍼레이터에 따라 계층적 구조를 선택하고 조회할 수 있도록 지원합니다.

접두사의 UTF-8 이진법 순서에 따르거나, 접두사를 지정하고 객체 키를 필터링한 리스트를 선택하여 단일 버킷의 모든 객체 키를 나열할 수 있습니다. 예를 들어 매개변수 `t` 를 추가하면 `tencent` 의 객체를 나열하고 `a` 또는 기타 부호를 접두사로 하는 객체는 건너뜁니다.

추가한 `/` 세퍼레이터에 따라 객체 키를 재구성할 수 있습니다. 접두사와 세퍼레이터를 결합하여 폴더 인덱스와 비슷한 기능을 구현할 수 있습니다.

Tencent Cloud COS에는 단일 버킷의 객체 수량 제한이 없으므로, 객체 키 리스트가 매우 방대합니다. 편리한 관리를 위해 단일 객체 나열 인터페이스는 객체 리스트를 최대 1000개까지 반환하며, 동시에 인디케이터를 반환하여 잘린 부분이 있는지 알려줍니다. 잘린 부분이 있는 경우, 다음 페이지에 객체 리스트가 존재함을 표시합니다. 인디케이터와 세퍼레이터를 바탕으로 객체 키 나열 요청을 여러 번 전송해 모든 객체 키를 나열하거나 필요한 콘텐츠를 찾을 수 있습니다.

사용 방법

COS 콘솔 사용

COS 콘솔을 이용해 객체를 검색할 수 있습니다. 자세한 내용은 객체 검색 콘솔 가이드 문서를 참조하십시오.

REST API 사용

REST API를 이용해 객체 키 리스트를 요청할 수 있습니다. 자세한 내용은 GET Bucket(List Objects) API 문서를 참조하십시오.

SDK 사용

직접 SDK의 객체 리스트 조회 방법을 호출할 수 있습니다. 자세한 내용은 아래 언어별 SDK 문서를 참조하십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- 미니프로그램 SDK

객체 복사
간편 복제

최종 업데이트 날짜: : 2023-03-14 14:44:27

적용 시나리오

복제 작업은 단일 요청에서 최대 5GB의 COS(Cloud Object Storage) 객체 사본을 작성합니다. 5GB가 넘는 객체를 복제하려면 멀티파트 복제 API를 사용하십시오. 복제 작업으로 다음을 수행할 수 있습니다.

- 객체의 복제본을 생성합니다.
- 객체를 복제하고 원래 객체를 삭제하여 객체 이름을 바꿉니다.
- 객체의 스토리지 클래스를 수정합니다. 소스 및 대상 모두로 동일한 객체 키를 선택하고 스토리지 클래스를 수정할 수 있습니다.
- 각각 다른 Tencent Cloud COS 리전에서 객체를 복제합니다.
- 객체 메타데이터를 수정합니다. 소스 및 대상 모두로 동일한 객체 키를 선택하고 객체 메타데이터를 수정할 수 있습니다.

복제 작업에서 원본 객체의 메타데이터는 기본적으로 상속되지만 생성 날짜는 대상 객체의 생성 날짜에 따릅니다.

> 설명 :
>
> - ARCHIVE 스토리지 클래스의 객체는 복제 및 붙여넣기가 지원되지 않습니다.
> - MAZ 버킷의 객체는 OAZ 버킷에 복제할 수 없습니다.
> - 서브 계정에는 PutObject, GetObject, GetObjectACL 객체 복제 권한이 부여되어야 합니다.

사용 방법

COS 콘솔 사용

COS 콘솔에서 객체를 복제합니다. 자세한 내용은 콘솔 가이드의 객체 복사를 참고하십시오.

REST API 사용

REST API를 사용해 객체 복제 요청을 시작합니다. 자세한 내용은 PUT Object - Copy를 참고하십시오.

SDK 사용

SDK에서 객체 복제 메소드를 직접 호출합니다. 자세한 내용은 아래의 해당 프로그래밍 언어에 대한 SDK 설명서를 참고하십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- 미니프로그램 SDK

멀티파트 복제

최종 업데이트 날짜: : 2022-12-28 15:53:32

적용 시나리오

5GB 이상의 객체를 복사해야 할 때, 파트 복사 방법을 선택해 구현해야 합니다. 멀티파트 업로드한 API를 사용해 신규 객체를 생성하고 Upload Part - Copy 기능을 사용하여 x-cos-copy-source 헤더를 가져와 원본 객체를 지정합니다. 과정은 다음과 같습니다.

1. 멀티파트 업로드한 객체를 초기화합니다.
2. 원본 객체의 데이터를 복사하면 x-cos-copy-source-range 헤더를 지정할 수 있으며, 한 번에 최대 5GB의 데이터만 복사할 수 있습니다.
3. 멀티파트 업로드를 완성합니다.

> 설명 :
> Tencent Cloud COS가 제공하는 SDK를 사용하면 블록 복사 기능을 쉽게 수행할 수 있습니다.

사용 방법

REST API 사용

REST API를 직접 사용해 블록 복사 요청을 실행할 수 있습니다. 다음 API 문서를 참조하십시오.

- Initiate Multipart Upload
- Upload Part - Copy
- Complete Multipart Upload
- Abort Multipart Upload

SDK 사용

SDK의 블록 복사 방법을 직접 호출할 수 있습니다. 아래 각 언어로 된 SDK 문서를 참조하십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET(C#) SDK
- Go SDK

- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- Mini Program SDK

객체 삭제
단일 객체 삭제

최종 업데이트 날짜: : 2021-06-29 17:25:13

적용 시나리오

Tencent Cloud COS는 객체 단일 삭제와 일괄 삭제를 지원합니다. 객체 단일 삭제 진행만 필요한 경우 객체 키로 API 호출 요청을 하여 삭제할 수 있습니다.

사용 방법

COS 콘솔 사용

COS 콘솔을 통해 객체를 삭제할 수 있습니다. 자세한 내용은 객체 삭제 콘솔 가이드 문서를 참조하십시오.

REST API 사용

REST API를 통해 즉각적으로 객체 단일 삭제 요청을 보낼 수 있습니다. 자세한 내용은 DELETE Object API 문서를 참조하십시오.

SDK 사용

SDK 호출을 통해 즉각적으로 객체 단일 삭제를 진행할 수 있습니다. 자세한 내용은 다음 언어별 SDK 문서를 참조하십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- 미니프로그램 SDK

여러 객체 삭제

최종 업데이트 날짜: : 2022-12-28 11:09:26

적용 시나리오

Tencent Cloud COS는 객체 일괄 삭제를 지원합니다. 콘솔, API, SDK 등 다양한 방식으로 객체 일괄 삭제를 진행할 수 있습니다.

삭제 작업이 성공적으로 이뤄진 경우 일반적으로 반환 내용이 없습니다. 오류가 발생할 경우 오류 정보를 반환합니다.

> 주의 :
> 1회당 최대 객체 1000개의 삭제 요청이 가능하며 더 많은 객체 삭제가 필요한 경우 리스트를 분할하여 요청을 보내십시오.

사용 방법

COS 콘솔 사용

COS 콘솔을 통해 여러 객체를 일괄 삭제할 수 있습니다. 자세한 내용은 객체 삭제 콘솔 가이드 문서를 참조하십시오.

REST API 사용

REST API를 통해 즉각적으로 객체 일괄 삭제 요청을 보낼 수 있습니다. 자세한 내용은 DELETE Multiple Objects API 문서를 참조하십시오.

SDK 사용

SDK 호출을 통해 즉각적으로 객체 일괄 삭제를 진행할 수 있습니다. 자세한 내용은 다음 언어별 SDK 문서를 참조하십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK

- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- Mini Program SDK

데이터 관리
라이프사이클 관리
라이프사이클 개요

최종 업데이트 날짜: : 2023-04-27 16:01:33

개요

COS(Cloud Object Storage)는 객체 기반의 라이프사이클 구성을 지원합니다. 라이프사이클 규칙을 사용하여 해당 객체에 대해 수행할 작업을 정의할 수 있습니다.

> 설명 :
> 버킷마다 최대 1,000개까지 라이프사이클 규칙을 추가할 수 있습니다.

사용 사례

로그 기록

라이프사이클이 구성되면 COS에 저장된 로그는 30일 후에 자동으로 보관되거나 2년 후에 삭제될 수 있습니다.

핫/콜드 데이터 티어링

핫 데이터는 업로드 후 짧은 시간 동안 자주 액세스하다가 일정 시간이 지나면 액세스가 거의 또는 전혀 이루어지지 않습니다. 따라서 30일 전에 업로드된 데이터는 STANDARD_IA 스토리지 클래스에, 60일 전에 업로드된 데이터는 ARCHIVE에 저장하도록 라이프사이클 규칙을 설정할 수 있습니다. 이 프로세스를 데이터 전환이라고 합니다.

아카이브 관리

파일 아카이브 관리에 COS를 사용하는 경우 금융, 의료 및 기타 산업의 컴플라이언스 요구 사항에 따라 파일의 모든 기존 버전을 장기간 저장해야 합니다. 이 경우 ARCHIVE 스토리지 클래스에서 파일의 기록 버전을 전환하고 저장하도록 라이프사이클을 구성할 수 있습니다.

구성 항목

라이프사이클 규칙을 생성하려면 다음 요소를 구성해야 합니다.

리소스

라이프사이클 실행 중에 히트할 데이터를 지정합니다. 범위 내에서 다루는 라이프사이클의 범위 및 데이터 유형을 사용자 정의할 수 있습니다. 라이프사이클 실행 중에 지정된 범위가 스캔되고 범위 내에서 구성된 데이터 유형에 대해 작업이 수행됩니다. 다음 규칙에 따라 범위를 지정할 수 있습니다.

- 접두사 지정: 디렉터리 이름 또는 파일 이름 접두사 매칭을 지원합니다.
- 태그 지정: 데이터를 태그로 필터링할 수 있습니다.

다음 데이터 유형을 구성할 수 있습니다.

- 현재 버전의 파일: 버킷의 최신 버전 객체입니다.
- 이전 버전의 파일: 버전 관리가 활성화된 후 저장된 이전 버전의 객체입니다. 버전 관리에 대한 자세한 내용은 버전 제어 개요를 참고하십시오.
- 삭제 마커: 객체가 삭제되었음을 나타내는 마커입니다. 라이프사이클 기능은 모든 기록 버전이 삭제된 후 마커를 자동으로 제거할 수 있습니다. 삭제 마커에 대한 자세한 내용은 삭제 마커를 참고하십시오.
- 조각 파일: 불완전한 멀티파트 업로드로 인해 생성된 조각입니다.

작업

객체가 히트될 때 수행할 작업:

- 데이터 전환: 지정된 기간 이후 객체를 STANDARD_IA, INTELLIGENT TIERING, ARCHIVE 또는 DEEP ARCHIVE로 전환합니다.
- 만료: 지정된 만료 시간이 지난 객체를 삭제합니다.

시간

상기 작업을 트리거하기 위한 시간 조건:
일 수 기준: 객체의 마지막 수정 날짜를 기준으로 객체에 정의된 작업을 수행할 시기를 지정할 수 있습니다.

사용 설명

> 설명 :
> 라이프사이클 사용 방법은 라이프사이클 설정을 참고하십시오.

규칙 시간 설명

파일 수정 시간

라이프사이클은 객체 수정 시간을 기반으로 트리거 규칙 실행을 지원합니다. PUT Object, PUT Object - Copy, POST Object 및 Complete Multipart Upload API와 같은 파일 쓰기 작업만 객체 수정 시간을 업데이트합니다. 라이프사이클을 기반으로 전환된 객체의 수정 시간은 업데이트되지 않습니다.

실행 일수 설명

규칙에서 정한 일수는 24시간을 기준으로 하며, 24시간 미만은 1일로 계산하지 않습니다.

예를 들어 1일 오후 3시에 파일을 수정하고 1일 후에 삭제되는 라이프사이클 규칙을 설정한 경우, 라이프 사이클 작업은 2일 0시에 파일 스캔을 시작하여, 2일 0시를 기준으로 최종 수정 시간이 1일을 넘은 파일에 대한 삭제 작업을 실행합니다. 1일 당일 업로드한 파일은 최종 수정 시간을 기준으로 1일을 초과하지 않았기 때문에 3일 0시까지 기다려야만 기록을 스캔하고 삭제를 실행할 수 있습니다.

최대 규칙 일수

라이프사이클은 최대 3650일까지 설정할 수 있습니다.

적용 시간

라이프사이클의 적용은 매일 스캔과 실행 두 가지 작업으로 나뉩니다.

- 스캔: COS는 현지 시간(GMT+8) 기준으로 매일 0시에 라이프사이클 규칙을 가져와 적용 범위 내의 모든 객체를 스캔합니다.
- 실행: 스캔에서 규칙이 지정한 날짜에 해당하는 객체가 스캔되면, 전환 또는 삭제 작업을 실행합니다.

예를 들어, 사용자가 2023년 1월 20일에 규칙 A를 설정하여 test.txt 파일의 수정 시간을 기준으로 10일 후에 삭제하도록 지정했다면, 2023년 1월 21일 0시부터 test.txt 파일의 수정 시간을 매일 0시에 스캔합니다. 만약 해당 파일의 마지막 수정 시간이 2023년 1월 15일인 경우, 2023년 1월 26일 0시에 실행되는 스캔 작업에서 파일이 삭제 조건을 충족시켰다고 판단되면, 스캔이 완료되면서 삭제 작업을 실행합니다.

> 주의 :
> 규칙 스캔과 실행 기간 중 규칙 상태를 변경하지 마십시오.. 변경하면 기존 규칙이 종료되어 전환 또는 삭제 작업이 올바르게 실행되지 않을 수 있습니다.

데이터 전환

단방향 원칙

데이터 전환은 단방향(STANDARD > STANDARD_IA > ARCHIVE 또는 STANDARD > ARCHIVE)이며 역방향으로 수행할 수 없습니다. PUT Object - Copy(비 ARCHIVE/DEEP ARCHIVE 전용) 또는 POST Object restore(ARCHIVE 및 DEEP ARCHIVE 전용)만 호출하여 colder 스토리지 클래스에서 hotter 스토리지 클래스로 데이터를 복구할 수 있습니다.

최종 일관성

동일한 객체 집합에 대해 여러 규칙이 구성되어 있고 서로 충돌하는 경우(만료 시 삭제 구성 제외) COS는 **객체를 coldest 스토리지 클래스로 전환**하는 규칙을 실행합니다.

예를 들어 규칙 A와 B가 각각 파일 수정 후 90일 후에 **객체를 STANDARD_IA로 전환** 및 **ARCHIVE로 전환**하도록 구성되어 있고 둘 다 동일한 객체 test.txt에 도달하면 규칙 B가 실행됩니다.

규칙	리소스	운영	시간 조건	실행
규칙 A	test.txt	객체를 STANDARD_IA 클래스로 전환	파일 수정 후 90일	규칙 충돌로 인해 실행 실패
규칙 B	test.txt	객체를 ARCHIVE 클래스로 전환	파일 수정 후 90일	실행 성공

주의 :

- COS에서 동일한 객체 세트에 대해 충돌하는 라이프사이클 규칙을 구성하지 않는 것이 좋습니다. 이로 인해 요금이 달라질 수 있습니다.
- 객체를 전환해도 객체가 업로드되거나 수정된 시간은 변경되지 않습니다.

만료 삭제

처리 로직

객체가 만료 시 지정된 삭제 라이프사이클 규칙과 일치하면 Tencent Cloud는 해당 객체를 비동기 삭제 큐에 추가합니다. 실제 삭제에는 약간의 지연이 있을 수 있습니다. GET 또는 HEAD 객체 작업을 수행하여 객체의 현재 상태를 가져올 수 있습니다.

최종 일관성

같은 그룹에 속한 객체에 여러 규칙을 설정하고, 충돌 상황이 존재할 경우, COS는 최단 기간의 만료 시간을 기준으로 실행하고, **만료 삭제의 실행 효과는 전환 스토리지 유형보다 큽니다.**

예를 들어 규칙 C와 D가 각각 파일 수정 180일 후 **객체를 STANDARD_IA로 전환**하고 **객체를 삭제**하도록 구성되어 있고 둘 다 동일한 객체 test.txt에 도달하면 규칙 D가 실행됩니다.

규칙	리소스	운영	시간 조건	실행
규칙 C	test.txt	객체를 STANDARD_IA 클래스로 전환	파일 수정 후 180일	규칙 충돌로 인해 실행 실패

규칙	리소스	운영	시간 조건	실행
규칙 D	test.txt	객체 삭제	파일 수정 후 180 일	실행 성공

주의 :
COS에서 동일한 객체 세트에 대해 충돌하는 라이프사이클 규칙을 구성하지 않는 것이 좋습니다. 이로 인해 요금이 달라질 수 있습니다.

비용 참고

수행 설명

- 라이프사이클 기능이 삭제 작업을 수행하면 백엔드 삭제 요청이 생성됩니다. 전환 작업을 수행하면 백엔드 삭제 및 쓰기 요청이 생성됩니다. 상기 작업으로 생성된 요청은 요청 청구서에 포함됩니다. 예를 들어 라이프사이클을 통해 STANDARD 스토리지 클래스의 'test.txt' 파일을 STANDARD_IA로 전환하면 두 개의 요청이 생성됩니다. 하나는 STANDARD의 데이터를 삭제하는 데 사용되고 다른 하나는 STANDARD_IA의 데이터를 쓰는 데 사용됩니다.

- 예외 상황이 발생하거나 버킷에 객체가 너무 많은 경우 라이프사이클 실행이 실패할 수 있습니다. 다른 이유로 인한 실패의 경우 GET 또는 HEAD Object 작업을 수행하여 현재 객체 상태를 가져옵니다.

- Tencent Cloud는 라이프사이클 실행이 완료되지 않으면 정확한 청구서를 제공할 수 없습니다.

기간 제한

- STANDARD_IA/INTELLIGENT TIERING, ARCHIVE 및 DEEP ARCHIVE 스토리지 클래스의 최소 스토리지 기간은 각각 30일, 90일 및 180일입니다. 전환 또는 삭제 작업 자체에 대해 추가 스토리지 요금이 발생하지 않습니다. COS는 30/90/180일 미만의 라이프사이클 구성을 무시하고 요청 시 올바른 구성만 수행합니다.

- 예를 들어 STANDARD_IA의 객체가 30일 이전에 전환되면 전환일부터 ARCHIVE 보관 요금이 발생하기 시작하고 30일까지 STANDARD_IA 보관 요금이 계속 부과됩니다. 또 다른 예는 보관된 객체가 90일 동안 저장되기 전에 만료 시 삭제되는 경우 90일까지 ARCHIVE 스토리지 요금이 계속 발생한다는 것입니다. DEEP ARCHIVE와 동일한 방식으로 작동합니다.

- **크기 제한**

  STANDARD_IA, ARCHIVE 및 DEEP ARCHIVE 스토리지 클래스에는 객체에 대한 최소 크기 제한이 있습니다. 예를 들어 64KB보다 작은 객체가 STANDARD_IA 스토리지 클래스에 업로드되면 64KB로 계산됩니다. 사용자 요금

을 줄이기 위해 라이프사이클 실행은 64KB보다 작은 객체의 스토리지 클래스를 전환하지 않습니다.

설명：
라이프사이클은 64KB보다 작은 객체의 스토리지 클래스를 전환하지 않습니다.

라이프사이클 구성 요소

최종 업데이트 날짜: : 2022-11-30 15:51:41

기본 구조

라이프사이클 설정은 XML 설명 방법을 사용합니다. 단일 또는 다중 라이프사이클 규칙을 설정할 수 있으며 기본 구조는 다음과 같습니다.

```
<LifecycleConfiguration>
<Rule>
<ID>**your lifecycle name**</ID>
<Status>Enabled</Status>
<Filter>
<And>
<Prefix>projectA/</Prefix>
<Tag>
<Key>key1</Key>
<Value>value1</Value>
</Tag>
</And>
</Filter>
**transition/expiration actions**
</Rule>
<Rule>
...
</Rule>
</LifecycleConfiguration>
```

모든 규칙은 다음 내용을 포함합니다.

- ID(선택 사항): 규칙의 내용을 나타내며 사용자 정의할 수 있습니다.
- Status: 규칙의 활성화 Enable 또는 비활성화 Disable 여부를 나타냅니다.
- Filter: 규칙이 적용되는 객체를 식별합니다.
- 작업: 위의 설명과 일치하는 객체에 대해 수행해야 하는 작업입니다.
- 시간: Days(객체가 마지막으로 수정된 날짜부터 계산됨) 또는 객체에 대해 수행된 작업을 기반으로 하는 Date입니다.

규칙 설명

Filter 요소

버킷의 모든 객체 대상

비어있는 필터링 조건을 지정하면 버킷의 모든 객체에 적용됩니다.

```
<LifecycleConfiguration>
<Rule>
<Filter>
</Filter>
<Status>Enabled</Status>
**transition/expiration actions**
</Rule>
</LifecycleConfiguration>
```

지정된 객체 키 접두사 대상

객체 접두사를 지정하면 접두사 설명에 부합한 일부 객체 그룹에 대해 작업을 수행할 수 있습니다. 예를 들어, logs/를 접두사로 한 모든 객체를 설정합니다.

```
<LifecycleConfiguration>
<Rule>
<Filter>
<Prefix>logs/</Prefix>
</Filter>
<Status>Enabled</Status>
**transition/expiration actions**
</Rule>
</LifecycleConfiguration>
```

지정된 객체 태그 대상

객체 태그에 부합한 key와 value를 필터링 조건으로 지정하고, 특정 태그의 객체에 대한 작업을 수행합니다. 예를 들어, 태그의 key=type과 value=image를 필터링 조건으로 한 객체를 설정합니다.

```
<LifecycleConfiguration>
<Rule>
<Filter>
<Tag>
<Key>type</Key>
<Value>image</Value>
</Tag>
</Filter>
<Status>Enabled</Status>
**transition/expiration actions**
</Rule>
</LifecycleConfiguration>
```

다중 필터링 조건 통합 사용

Tencent Cloud COS(Cloud Object Storage)는 AND의 로직을 통해 다중 필터링 조건을 통합 사용할 수 있습니다. 예를 들어, logs/를 접두사로 하면서 객체 태그의 key=type과 value=image를 필터링 조건으로 한 객체를 설정합니다.

```
<LifecycleConfiguration>
<Rule>
<Filter>
<And>
<Prefix>logs/</Prefix>
<Tag>
<Key>type</Key>
<Value>image</Value>
</Tag>
</And>
</Filter>
<Status>Enabled</Status>
**transition/expiration actions**
</Rule>
</LifecycleConfiguration>
```

작업 요소

라이프사이클 규칙 중, 조건에 부합한 한 그룹 객체에 단일 또는 다중 작업을 수행할 수 있습니다.

전환 작업

한 스토리지 유형에서 다른 스토리지 유형으로 객체를 전환하려면 Transition 작업을 지정합니다. 버전이 지정된 버킷의 경우 전환 작업이 현재 객체 버전에 적용됩니다. Transition 날짜를 0일로 짧게 설정할 수 있습니다. 예를 들어 30일 후에 객체를 아카이브 스토리지로 전환합니다.

```
<Transition>
<StorageClass>ARCHIVE</StorageClass>
<Days>30</Days>
</Transition>
```

만료 삭제

만료된 객체를 삭제하려면 Expiration 작업을 지정합니다. 버전이 지정되지 않은 버킷의 경우 만료된 객체는 영구적으로 삭제됩니다. 버전이 지정된 버킷의 경우 만료된 객체가 **DeleteMarker로 추가**되고 현재 버전이 됩니다. 예를 들어 만료일이 30일인 객체를 삭제합니다.

```
<Expiration>
<Days>30</Days>
</Expiration>
```

미완성 멀티파트 업로드

> 주의 :
> 동일한 라이프사이클 규칙에서 객체 태그 지정과 업로드 완료되지 않은 파일 조각 정리를 동시에 설정할 수 없습니다.

미리 정의된 기간 내에 성공적으로 완료되지 않은 경우 특정 UploadId가 있는 멀티파트 업로드 작업을 삭제하려면 AbortIncompleteMultipartUpload 작업을 지정하십시오. 이 경우 이러한 작업도 재개하거나 인덱싱할 수 없습니다. 예를 들어, 7일 이내에 성공적으로 완료되지 않은 멀티파트 업로드 작업을 중단합니다.

```
<AbortIncompleteMultipartUpload>
<DaysAfterInitiation>7</DaysAfterInitiation>
</AbortIncompleteMultipartUpload>
```

현재 버전이 아닌 객체

버전이 지정된 버킷에서 전환 작업은 최신 버전의 객체에만 적용되고 만료 작업은 객체에 삭제 마커를 추가하는 결과만 가져옵니다. 따라서 COS는 현재 버전이 아닌 객체에 대해 다음 작업을 제공합니다.

지정된 시간에 현재 버전의 객체를 다른 스토리지 클래스로 전환하려면 NoncurrentVersionTransition 작업을 지정합니다. 예를 들어, 30일 후에 객체의 이전 버전을 아카이브 스토리지로 전환합니다.

```
<NoncurrentVersionTransition>
<StorageClass>ARCHIVE</StorageClass>
<Days>30</Days>
</NoncurrentVersionTransition>
```

지정된 시간에 만료된 현재 버전의 객체를 삭제하려면 NoncurrentVersionExpiration 작업을 지정합니다. 예를 들어 30일 후에 만료된 이전 버전의 객체를 삭제합니다.

```
<NoncurrentVersionExpiration>
<Days>30</Days>
</NoncurrentVersionExpiration>
```

ExpiredObjectDeleteMarker를 지정하여 나머지 삭제 마커를 제거합니다. 입력 가능 값은 true 또는 false입니다. 이 옵션의 적용은 만료된 기록 버전(NoncurrentVersionExpiration) 제거 작업에 의해 트리거됩니다. 이 기능이 활성화되어 있으면 객체의 마지막 기록 버전이 라이프사이클에 의해 삭제될 때 나머지 여러 삭제 마커가 자동으로 제거됩니다.

만료된 기록 버전(NoncurrentVersionExpiration) 제거 작업이 적용되는 구체적인 상황은 다음과 같습니다.

- ExpiredObjectDeleteMarker 활성화 여부와 상관없이 삭제 마커가 하나만 남아 있으면 마지막 삭제 마커가 자동으로 정리됩니다.
- 남은 삭제 마커가 여러 개인 경우 COS는 ExpiredObjectDeleteMarker가 활성화되어 있는지 확인합니다. true이면 나머지 여러 삭제 마커가 자동으로 정리되고, false이면 나머지 여러 삭제 마커가 정리되지 않습니다.

```
<ExpiredObjectDeleteMarker>
<Days>true|false</Days>
</ExpiredObjectDeleteMarker>
```

시간 요소

일별 계산

Days는 객체가 마지막으로 수정된 이후의 일 수를 나타냅니다.

- 예를 들어 객체가 0일 후 아카이브 스토리지로 전환되도록 설정되어 있는 경우 객체가 2018-01-01 23:55:00 GMT+8에 업로드되면 2018-01-02 00:00:00 GMT+8에 전환 대기열에 추가되고 2018-01-02 23:59:59 GMT+8 이전에 전환됩니다.
- 예를 들어 객체가 만료로 인해 1일 후에 삭제되도록 설정되어 있는 경우 객체가 2018-01-01 23:55:00 GMT+8에 업로드되면 객체가 2018-01-03 00:00:00 GMT+8에 만료 대기열에 추가되고 2018-01-03 23:59:59 GMT+8 이전에 삭제됩니다.

지정 날짜 기준

특정 Date의 필터 기준에 따라 모든 규정된 개체에 대해 미리 정의된 작업을 수행합니다. 날짜 값은 ISO8601 형식을 따라야 합니다. 시간은 항상 00:00 GMT+8입니다.
예를 들어, 2018년 01월 01일은 2018-01-01T00:00:00+08:00로 사용합니다.

라이프사이클 설정

최종 업데이트 날짜: : 2022-12-29 10:38:03

적용 시나리오

라이프사이클 설정을 통해 규칙에 부합한 객체가 지정된 조건 하에 일부 조작을 자동 수행할 수 있습니다. 예시:

- 스토리지 유형 전환: 생성된 객체를 지정 시간 후 표준IA 스토리지 유형(STANDARD_IA), INTELLIGENT TIERING(INTELLIGENT_TIERING), CAS(ARCHIVE) 유형 및 DEEP ARCHIVE로 전환할 수 있습니다.
- 만료 삭제: 객체의 만료시간을 설정하면 객체가 만료된 후 자동으로 삭제됩니다.

자세한 내용은 라이프사이클 개요 문서 및 라이프사이클 설정 요소 문서를 참조하십시오.

사용 방법

COS 콘솔 사용

COS 스토리지 콘솔을 사용해 라이프사이클을 설정할 수 있습니다. 자세한 내용은 라이프사이클 설정 콘솔 가이드 문서를 참조해 주십시오.

REST API 사용

REST API를 직접 사용해 버킷 중 객체의 라이프사이클을 설정 및 관리할 수 있습니다. 자세한 내용은 다음 API 문서를 참조해 주십시오.

- PUT Bucket lifecycle
- GET Buket lifecycle
- DELETE Bucket lifecycle

SDK 사용

SDK의 라이프사이클을 직접 호출할 수 있습니다. 자세한 내용은 아래 각 언어로 된 SDK 문서를 참조해 주십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK

- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK
- Mini Program SDK

정적 웹 사이트 호스팅

최종 업데이트 날짜: : 2023-01-06 16:22:53

기본 개념

정적 웹 사이트는 정적 콘텐츠(예: HTML) 또는 클라이언트 스크립트를 포함하는 웹 사이트입니다. 콘솔을 통해 사용자 지정 도메인 이름이 있는 버킷에 대한 정적 웹 사이트를 구성할 수 있습니다. 동적 웹 사이트에는 PHP, JSP 또는 ASP.NET과 같은 서버 스크립트가 포함되어 있으며 서버에서 처리되어야 합니다. Tencent Cloud COS(Cloud Object Storage)에서 정적 웹사이트를 호스팅할 수 있지만 서버 스크립트를 작성할 수는 없습니다. 동적 웹 사이트 배포의 경우 서버 코드 배포용 CVM(Cloud Virtual Machine)을 사용하는 것이 좋습니다.

예시

사용자가 examplebucket-1250000000라는 이름의 버킷을 생성하여 다음과 같은 파일을 업로드하였습니다.

```
index.html
404.html
403.html
test.html
docs/a.html
images/
```

정적 웹 사이트

활성화 전: 다음 기본 액세스 도메인으로 버킷에 액세스하여 다운로드 알림을 팝업하면 `index.html` 파일을 로컬에 저장할 수 있습니다.

```
https://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com/index.html
```

활성화 후: 다음 액세스 노드로 버킷에 액세스하면 브라우저에서 `index.html` 의 페이지 콘텐츠를 조회할 수 있습니다.

```
https://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com/index.html
```

강제 HTTPS

활성화 전: HTTP로부터 요청이 온 경우 노드 URL에 액세스하여 HTTP의 암호화가 되지 않은 전송 프로토콜을 유지합니다.

```
http://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com
```

활성화 후: HTTP나 HTTPS로부터 요청이 온 경우 노드에 액세스하여 HTTPS의 암호화 된 전송 프로토콜을 계속 유지합니다.

```
https://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com
```

인덱스 문서

정적 웹 사이트의 첫 페이지인 인덱스 문서는 사용자가 웹 페이지의 루트 디렉터리 혹은 서브 디렉터리에 요청을 보낼 때 반환되는 웹 페이지로 `index.html` 로 불립니다.
사용자가 버킷 액세스 도메인(예시로 `https://examplebucketbucket-1250000000.cos-website.ap-guangzhou.myqcloud.com`)으로 정적 웹 사이트에 액세스할 경우 특정 웹 페이지가 요청되지 않으며 웹 서버는 첫 화면을 반환합니다.

사용자가 버킷의 임의 디렉터리(루트 디렉터리를 포함)에 액세스하고 URL 주소가 `/` 로 끝나면 자동으로 그 디렉터리 인덱스 문서가 먼저 매칭됩니다. URL의 `/` 은 선택 사항이며 아래 URL 모두 인덱스 문서를 반환합니다.

```
http://www.examplebucket.com/
http://www.examplebucket.com
```

> 주의 :
> 버킷에 폴더가 생성되면 폴더의 각 레벨에 인덱스 파일을 추가해야 합니다.

오류 문서

오류 문서를 설정하기 전이라고 가정한 후 아래 페이지에 액세스하면 404 상태 코드가 반환되며 페이지에 기본 오류 페이지 정보가 보입니다.

```
https://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com/webpage.html
```

오류 문서를 설정한 후 아래 페이지에 액세스하면 똑같이 404 상태 코드가 반환되지만 페이지에 특정 오류 페이지 정보가 보입니다.

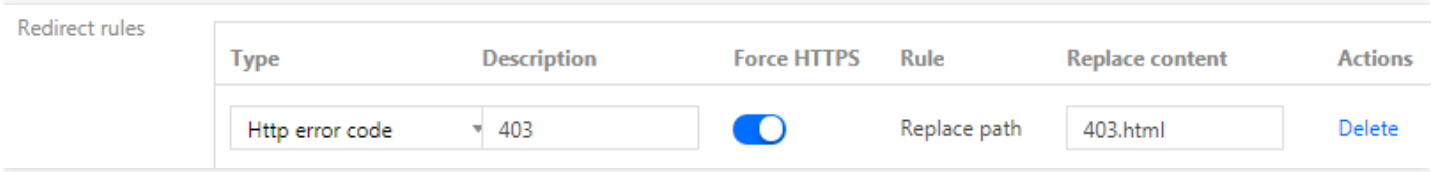
```
https://examplebucket-1250000000.cos-website.ap-guangzhou.myqcloud.com/webpage.html
```

리디렉션 규칙

설명：
호스팅된 정적 웹 사이트에 대한 리디렉션 규칙을 구성하려면 대체 파일의 경로가 버킷의 객체 경로여야 합니다.

에러 코드 리디렉션 설정하기

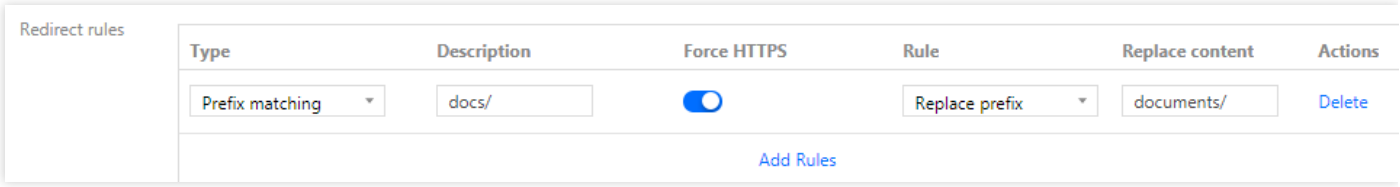
webpage.html 문서로 **개인 읽기/쓰기** 공개 액세스 권한을 설정하고 사용자가 이 문서에 액세스할 경우 403 오류가 반환됩니다.
403 에러 코드가 403.html에 리디렉션되면 브라우저는 403.html 콘텐츠를 반환합니다.
403.html 문서를 설정하지 않으면 브라우저는 오류 문서 혹은 기본 오류 정보를 반환합니다.



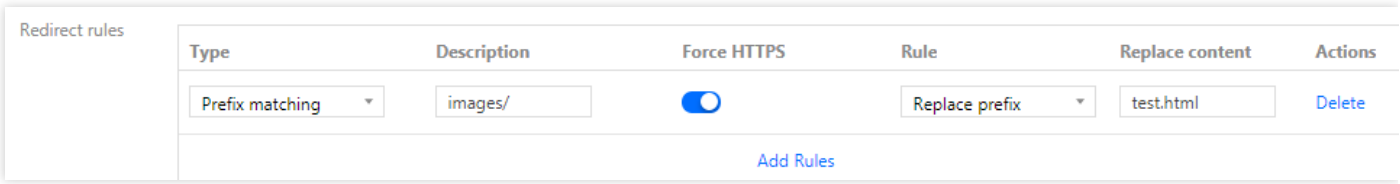
접두사 매칭 설정

주의：
접두사 매칭은 와일드카드를 지원하지 않습니다. 접두사가 index1/ 및 index2/인 두 폴더를 리디렉션하려면 `index*/` 를 매칭 규칙으로 사용할 수 없으며 각각 해당하는 매칭 규칙을 생성해야 합니다.

1. 폴더 이름을 `docs/` 에서 `documents/` 로 변경한 후 사용자가 `docs/` 폴더에 액세스하면 오류가 발생하므로 접두사 `docs/` 요청을 `documents/` 에 리디렉션 하십시오.



2. `images/` 폴더를 삭제하면(즉 접두사 `images/` 를 가진 모든 객체를 삭제), 리디렉션 규칙을 추가하여 접두사 `images/` 를 가진 임의의 객체 요청을 `test.html` 페이지에 리디렉션할 수 있습니다.



인벤토리 개요

최종 업데이트 날짜: : 2023-01-18 15:45:03

인벤토리가 무엇인가요?

인벤토리는 사용자가 버킷의 객체를 관리하도록 도와주는 기능입니다. COS를 대신해 List API 동기화 작업을 주기적으로 할 수 있습니다. Cloud Object Storage(COS)는 사용자의 인벤토리 작업 설정에 따라 매일 혹은 매주 제 시간에 사용자 버킷 내의 지정된 객체나 똑같은 접두사를 가진 객체를 스캐닝하고 인벤토리 보고서를 CSV 형식 파일로 출력하여 사용자가 지정한 버킷에 저장합니다. 파일에서 저장한 객체와 이에 대응되는 메타데이터 목록을 만들 수 있으며 사용자의 설정 정보에 따라 사용자가 필요한 객체 속성 정보를 기록합니다.

인벤토리 기능의 용도는 다음을 포함하며 이에 국한되지 않습니다.

- 객체 복사 및 암호화 상태를 심의하고 보고합니다.
- 비즈니스 작업 흐름 및 빅 데이터 작업을 간소화하고 가속화합니다.

> 주의 :
>
> - 사용자는 버킷에서 여러 개의 인벤토리 작업을 설정할 수 있으며 인벤토리 작업을 실행하는 동안 객체의 메타데이터 등의 속성 정보는 스캐닝할 수 있지만 객체 콘텐츠를 읽어 들일 수는 없습니다.

인벤토리 매개변수

사용자가 인벤토리 작업을 설정한 후 COS는 설정에 따라 정기적으로 사용자 버킷 내에 지정된 객체를 스캐닝하며 CSV 형식 파일로 인벤토리 보고서를 출력합니다. 현재 COS 인벤토리 보고서는 다음 정보를 기록할 수 있습니다.

인벤토리 정보	설명
AppID	계정 ID
Bucket	인벤토리 작업을 실행할 버킷 이름
fileFormat	파일 형식
listObjectCount	나열된 객체 수량. 이 항목에 따라 요금이 부과되며, 자세한 사항은 관리 기능 요금의 인벤토리 기능 요금 부분을 참고하십시오.
listStorageSize	나열된 객체 크기

인벤토리 정보	설명
filterObjectCount	선별된 객체 수
filterStorageSize	선별된 객체 크기
Key	버킷의 객체 파일 이름. CSV 파일 형식을 사용할 때 객체 파일 이름은 URL 코드 형식으로 복호화한 후 사용할 수 있습니다.
VersionId	객체 버전 ID. 버킷에서 버전 제어를 활성화한 후 COS는 버킷에 추가된 객체에 버전 번호를 지정할 수 있습니다. 인벤토리가 객체의 현재 버전 전용인 경우 해당 필드는 포함하지 않습니다.
IsLatest	객체가 최신 버전이라면 True로 설정합니다. 인벤토리가 객체의 현재 버전 전용인 경우 해당 필드는 포함하지 않습니다.
IsDeleteMarker	객체가 삭제 마커라면 True로 설정합니다. 인벤토리가 객체의 현재 버전 전용인 경우 해당 필드는 포함하지 않습니다.
Size	객체 크기(바이트 단위)
LastModifiedDate	객체의 최근 수정 일자(늦은 일자 기준)
ETag	실물 태그는 객체의 해시 값입니다. ETag는 객체 콘텐츠의 변경 내용만 반영되며 객체 메타데이터의 변경 내용은 반영되지 않습니다. ETag는 객체 데이터 MD5의 요약일 수도 있고 아닐 수도 있습니다. 이 여부는 객체의 생성 방식과 암호화 방식으로 결정되지 않습니다.
StorageClass	객체의 스토리지 분류. 자세한 내용은 스토리지 유형을 참고하십시오.
IsMultipartUploaded	객체가 멀티파트 업로드의 형식으로 업로드되면 True로 설정합니다. 자세한 내용은 멀티파트 업로드를 참고하십시오.
Replicationstatus	객체 복제에서 소스 및 복제본 파일의 상태입니다. 소스 파일 상태: PENDING`(복제 예정), COMPLETED(복제됨) 또는 FAILED(복제 실패); 복제 파일 상태: REPLICA(생성된 복제 파일로 복제됨). 자세한 내용은 복사 작용 설명을 참고하십시오.
Tag	객체 태그

인벤토리 설정 방법

인벤토리를 설정하기 전 두 가지 개념을 알아야 합니다.

- 소스 버킷: 인벤토리 기능을 활성화하고 싶은 버킷.
  - 인벤토리가 나열한 객체를 포함합니다.

  - 인벤토리 설정을 포함합니다.
- 타깃 버킷: 스토리지 인벤토리의 버킷.
  - 인벤토리 목록 파일을 포함합니다.
  - Manifest 파일을 포함하며 인벤토리 목록 파일 위치를 설명합니다.

다음은 인벤토리 설정 절차입니다.

소스 버킷에서 분석할 객체 정보 지정

COS에 어떤 객체 정보 분석이 필요한지 보고해야 합니다. 따라서 인벤토리 기능을 설정할 때 소스 버킷에서 아래 정보를 설정해야 합니다.

- 객체 버전 선택: 모든 객체 버전을 나열하거나 현재 버전만 나열합니다. 모든 객체 버전 나열을 선택했다면 COS는 동일 이름을 가진 객체의 지난 모든 버전을 인벤토리 보고서에 나열합니다. 현재 버전만 나열하도록 선택했다면 COS는 최신 버전 객체만 기록합니다.
- 분석이 필요한 객체 속성 설정: 객체 속성 중 어떤 정보가 인벤토리 보고서에 어떤 정보가 기록되는지 COS에 알려야 하며 현재 지원하는 객체 속성은 계정 ID, 소스 버킷 이름, 객체 파일 이름, 객체 버전 ID, 최신 버전 여부, 삭제 마커 여부, 객체 크기, 객체 최신 수정 일자, ETag, 객체 스토리지 종류, 리전 간 복제 표시, 멀티파트 업로드 파일에 속하는지의 여부가 포함됩니다.

인벤토리 보고서의 스토리지 정보 설정

COS는 어느 주파수에 따라 인벤토리 보고서를 내보낼지 공지하며 어느 버킷에 인벤토리 보고서가 저장되고 암호화 여부가 필요한지 결정합니다. 설정이 필요한 정보는 다음과 같습니다.

- 인벤토리 내보내기 빈도수 선택: 매일 혹은 매주. 이 설정을 통해 COS에 어떤 빈도수에 따라 인벤토리 기능을 실행할지 알릴 수 있습니다.
- 인벤토리 암호화 선택: 비암호화 혹은 SSE-COS. SSE-COS 암호화를 선택했다면 생성한 인벤토리 보고서에 대해 암호화를 진행합니다.
- 인벤토리를 내보낼 위치 설정: 버킷 저장이 필요한 인벤토리 보고서를 지정해야 합니다.

> 주의 :
> 타깃 버킷은 반드시 소스 버킷과 동일 리전에 위치해야 하며 양자는 동일 버킷일 수 있습니다.

사용 방법

콘솔로 인벤토리 설정

인벤토리 기능 활성화 콘솔 문서를 참고하여 콘솔을 통한 인벤토리 기능 설정 방법을 알 수 있습니다.

API로 인벤토리 설정

API를 사용해 지정 버킷에 인벤토리 기능을 활성화하려면 다음 단계를 참고하십시오.

1. COS 역할을 생성합니다.
2. COS 역할 권한을 바인딩합니다.
3. 인벤토리 기능을 활성화합니다.

1. COS 역할 생성

COS 역할 생성과 구체적인 인터페이스 정보는 CreateRole을 참고하십시오.
roleName은 반드시 COS_QcsRole이어야 합니다.
policyDocument:

```
{
"version": "2.0",
"statement": [{
"action": "name/sts:AssumeRole",
"effect": "allow",
"principal":{
"service": "cos.cloud.tencent.com"
}
}]
}
```

2. COS 역할 권한 바인딩

역할 권한 바인딩 권한과 구체적인 인터페이스 정보는 AttachRolePolicy를 참고하십시오.
policyName은 QcloudCOSFullAccess, roleName은 1단계의 COS_QcsRole입니다. roleName 생성 시 반환된 roleID를 사용할 수 있습니다.

3. 인벤토리 기능 활성화

인터페이스 호출, 인벤토리 기능 활성화 및 구체적인 인터페이스 정보는 PUT Bucket inventory를 참고하십시오. 매니페스트 파일에 저장할 타깃 버킷과 소스 버킷은 같은 리전에 있어야 합니다.

인벤토리 보고서 스토리지 경로

인벤토리 보고서와 Manifest 관련 파일은 타깃 버킷에 배포될 수 있으며 그 중 인벤토리 보고서는 다음 경로로 배포될 수 있습니다.

```
destination-prefix/appid/source-bucket/config-ID/
```

Manifest 관련 파일은 다음 경로로 타깃 버킷에 배포됩니다.

```
destination-prefix/appid/source-bucket/config-ID/YYYYMMDD/manifest.json
destination-prefix/appid/source-bucket/config-ID/YYYYMMDD/manifest.checksum
```

경로에서 나타내는 의미는 다음과 같습니다.

- **desitination-prefix**: 사용자가 인벤토리를 설정할 때 설치하는 '타깃 접두사'입니다. 타깃 버킷에서 공용 위치에 있는 모든 인벤토리 보고서를 그룹화하는데 사용할 수 있습니다.
- **source-bucket**: 인벤토리 보고서와 일치하는 소스 버킷 이름입니다. 이 폴더는 각각의 인벤토리 보고서를 여러 소스 버킷에서 동일한 타깃 버킷에 발송할 때 발생하는 충돌을 방지하기 위함입니다.
- **config-ID**: 사용자가 인벤토리를 설정할 때 설치된 '인벤토리 이름'으로 동일한 소스 버킷이 여러 인벤토리 보고서를 설치하고 이를 동일한 타깃 버킷에 발송할 때 config-ID로 상이한 인벤토리 보고서를 구분할 수 있습니다.
- **YYYYMMDD**: 타임스탬프. 인벤토리 보고서를 생성할 때 버킷이 스캐닝하기 시작하는 시간과 일자를 포함합니다.
- **manifest.json**: Manifest 파일을 가리킵니다.
- **manifest.checksum**: manifest.json 파일 콘텐츠의 MD5를 가리킵니다.

그 중 Manifest와 관련된 파일은 총 2개의 파일로 manifest.json과 manifest.checksum 입니다.

> 설명 :
>
> Mainfest 관련 파일 소개는 다음과 같습니다.
>
> - manifest.json과 manifest.chenksum은 모두 Manifest 파일입니다. manifest.json은 인벤토리 보고서의 위치를 설명하고 manifest.checksum은 manifest.json 파일 콘텐츠의 MD5입니다. 새로운 인벤토리 보고서를 딜리버리할 때마다 새로운 Manifest 파일이 따라 나올 수 있습니다.
> - manifest.json이 포함하는 모든 Manifest는 인벤토리의 메타데이터와 관련된 기타 기본 정보를 제공하며 그 정보는 아래와 같습니다.
>   - 소스 버킷 이름.
>   - 타깃 버킷 이름.
>   - 인벤토리 버전.
>   - 타임스탬프. 인벤토리 보고서 생성 시 버킷 일자와 시간을 스캐닝하기 시작합니다.
>   - 매니페스트 파일의 형식과 구성.
>   - 타깃 버킷 인벤토리가 보고하는 객체 키, 크기, md5Checksum.

다음은 CSV 형식 인벤토리의 manifest.json 파일에 있는 Manifest 예시입니다.

```
{
"sourceAppid": "1250000000",
```

```
"sourceBucket": "example-source-bucket",
"destinationAppid": "1250000000",
"destinationBucket": "example-inventory-destination-bucket",
"fileFormat": "CSV",
"listObjectCount": "13",
"listStorageSize": "7212835",
"filterObjectCount": "13",
"filterStorageSize": "7212835",
"fileSchema": "Appid, Bucket, Key, Size, LastModifiedDate, ETag, StorageClass, Is
MultipartUploaded, ReplicationStatus",
"files": [
{
"key": "cos_bucket_inventory/1250000000/examplebucket/inventory01/04d73d9debc73d9
f0bf85af461abde6c.csv.gz",
"size": "502",
"md5Checksum": "7d40288a09c25b302ad6cb5fced54f35"
}
]
}
```


인벤토리의 일치성

COS 인벤토리 보고서는 새 객체와 덮어쓴 PUT 그리고 삭제의 최종 일관성을 제공합니다. 따라서 인벤토리 보고서에는 최근 추가되거나 삭제한 객체가 포함되지 않을 수 있습니다. 예시로 COS가 사용자 설정의 인벤토리 업무 과정을 실행할 때 사용자가 객체를 업로드하거나 삭제할 경우 이 작업 결과는 인벤토리 보고서에 반영되지 않을 수 있습니다.

객체가 작업을 실행하기 전 객체의 상태를 확인하려면 HEAD Object API를 사용하여 객체 메타데이터를 인덱스하거나 COS 콘솔에서 객체 속성을 조회하시기 바랍니다.

버킷 태그 개요

최종 업데이트 날짜: : 2022-08-24 11:53:40

개요

버킷 태그는 키 값 쌍(key = value)입니다. 태그 키(Key), 태그 값(Value), “=”로 이루어집니다. 예시 group = IT. 버킷을 관리하는 표시로서 사용자가 버킷을 그룹 관리하는 경우 편리합니다. 지정 버킷에 태그 설정, 조회, 삭제 작업을 할 수 있습니다.

규격 및 제한

태그 키 제한

`qcs:` , `project` , `항목` 등을 첫 글자로 시작하는 태그 키는 시스템에 미리 남겨진 태그 키이며 시스템에 미리 남겨진 태그 키는 생성할 수 없습니다.

- UTF-8 포맷은 문자 부호, 빈칸, 숫자, 특수 문자 `+ - = ._ : / @` 로 나타낼 수 있습니다.
- 태그 키 길이는 문자 부호 0 - 127개까지 허용합니다(UTF-8 포맷 적용).
- 태그 키는 영어 대소문자를 구별합니다.

태그 값 제한

- UTF-8 포맷은 문자 부호, 빈칸, 숫자, 특수 문자 `+ - = ._ : / @` 로 나타낼 수 있습니다.
- 태그 값 길이는 문자 부호 0 - 255개까지 가능합니다.(UTF-8 포맷 적용)
- 태그 값은 영어 대소문자를 구분합니다.

태그 수 제한

- 버킷의 경우: 리소스 하나당 서로 다른 버킷 태그 최대 50개까지 가능합니다.
- 태그의 경우:
- 사용자 한 명이 서로 다른 키를 최대 1,000개까지 사용할 수 있습니다.
- 하나의 key는 최대 1,000개 값을 가집니다.
- 동일한 버킷에서는 똑같은 key를 여러 개 사용할 수 없습니다.

사용 방법

콘솔, API를 통해 버킷 태그를 설정할 수 있습니다.

COS 콘솔 사용

COS 콘솔을 사용해 버킷 태그를 설정해야 할 경우 버킷 태그 설정 콘솔 가이드 문서를 참조하십시오.

REST API 사용

다음 API를 통해 버킷 태그를 관리할 수 있습니다.

- PUT Bucket tagging
- GET Bucket tagging
- DELETE Bucket tagging

객체 태그 개요

최종 업데이트 날짜: : 2022-08-08 14:52:35

기능 개요

객체 태그 기능은 객체에 키 값 쌍을 식별자로 추가하여, 사용자가 버킷에 있는 객체를 그룹화하고 관리하는 데 도움이 되도록 설계되었습니다. 객체 태그는 태그 키(`tagKey`)와 태그 값(`tagValue`), 그리고 `=` 로 연결되어 구성됩니다(예: `group=IT`). 사용자는 지정 객체의 태그 설정, 조회, 삭제 작업을 할 수 있습니다.

> 주의 :
> 객체 태그 기능은 과금 항목이며, 자세한 가격은 제품 가격 문서를 참조하십시오.

사용 방법

COS 콘솔 사용

Cloud Object Storage(COS) 콘솔로 객체 태그를 관리할 경우 자세한 정보는 객체 태그 설정을 참고하십시오.

REST API 사용

다음 API를 통해 객체 태그를 관리할 수 있습니다.

- PUT Object tagging
- GET Object tagging
- DELETE Object tagging

규격 및 제한

태그 키 제한

- UTF-8 포맷은 문자 부호, 빈칸, 숫자[0-9], 특수 문자 `+ - = ._ : / @` 로 나타낼 수 있습니다.
- 태그 키 길이는 문자 부호 1-127개까지 가능합니다.(UTF-8 포맷 적용)
- 태그 키는 영어 대소문자를 구별합니다.

태그 값 제한

- UTF-8 포맷은 문자 부호, 빈칸, 숫자[0-9], 특수 문자 `+ - = ._ : / @` 로 나타낼 수 있습니다.

- 태그 값 길이는 문자 부호 1-255개까지 허용합니다.(UTF-8 포맷 적용)
- 태그 값은 영어 대소문자를 구분합니다.

태그 수 제한

- 객체의 경우: 객체 하나당 상이한 객체 태그 최대 10개까지 가능합니다.
- 태그의 경우: 제한이 없습니다.

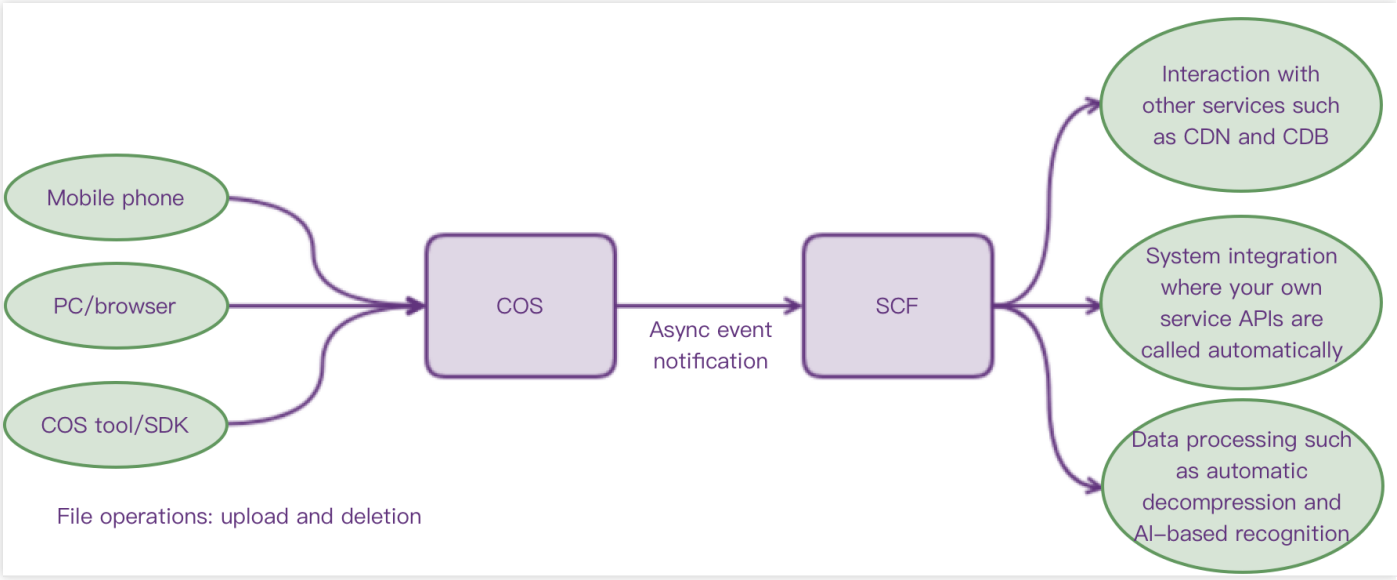
이벤트 알림

최종 업데이트 날짜: : 2022-05-17 16:17:26

개요

COS 리소스에 변동이 생긴 경우 (예를 들어 새 파일 업로드, 파일 삭제) 곧바로 공지 메세지를 받을 수 있습니다. 이벤트 공지는 SCF(Serverless Cloud Function)와 결합하여 더욱 많은 응용 시나리오를 만들어 냅니다.

- **제품간 연동**: 새 파일을 COS에 업로드하면 CDN 캐시가 자동으로 새로 고침됩니다. 새 파일을 COS에 업로드하면 자동으로 데이터베이스가 갱신됩니다.
- **시스템 통합**: COS 상의 파일에 변경(새로 만들기, 삭제, 덮어쓰기)이 생긴 경우 자동으로 서비스 인터페이스를 호출합니다. UGC(User Generated Content) 시나리오에서 이벤트 공지 기능을 기반으로 모바일과 서버가 연동됩니다.
- **데이터 프로세스**: COS 파일은 자동으로 프로세스됩니다. 자동 압축 해제, AI 인식 등을 예로 들 수 있습니다.



COS 이벤트 공지는 다음과 같은 특징을 가집니다.

- 비동기화 프로세스: 공지 발송은 COS의 정상 작업에 영향을 주지 않습니다.
- 공지 타깃: 리전 내 SCF 함수로의 공지만 발송합니다.

현재 다음 COS 이벤트를 지원합니다.

이벤트 유형	설명

이벤트 유형	설명
cos:ObjectCreated:*	아래 언급한 모든 업로드 이벤트는 SCF를 트리거할 수 있습니다.
cos:ObjectCreated:Put	PUT Object 인터페이스를 사용해 파일을 생성할 때 SCF를 트리거합니다.
cos:ObjectCreated:Post	POST Object 인터페이스를 사용해 파일을 생성할 때 SCF를 트리거합니다.
cos:ObjectCreated:Copy	PUT Object - Copy 인터페이스를 사용해 파일을 생성할 때 SCF를 트리거합니다.
cos:ObjectCreated:CompleteMultipartUpload	Complete Multipart Upload 인터페이스를 사용해 파일을 생성할 때 SCF를 트리거합니다.
cos:ObjectCreated:Origin	이미지 Origin-pull이 발생한 경우 SCF를 트리거합니다.
cos:ObjectCreated:Replication	리전 간 복제로 객체를 생성할 경우 SCF를 트리거합니다.
cos:ObjectRemove:*	아래 언급한 모든 삭제 이벤트는 SCF를 트리거할 수 있습니다.
cos:ObjectRemove:Delete	버전 제어 버킷을 활성화하지 않고 DELETE Object 인터페이스로 객체를 삭제하거나 versionid를 사용해 지정된 버전의 객체를 삭제할 경우 SCF를 트리거합니다.
cos:ObjectRemove:DeleteMarkerCreated	활성화되어 있거나 잠시 중지된 버전이 제어하는 버킷에서 DELETE Object 인터페이스로 객체를 삭제할 경우 SCF를 트리거 합니다.
cos:ObjectRestore:Post	보관 복구 작업이 생성될 때 SCF를 트리거합니다.
cos:ObjectRestore:Completed	보관 복구 작업이 완료되었을 때 SCF를 트리거합니다.

COS 이벤트 공지 사용 방법

다음은 COS 이벤트 공지 사용 절차입니다.

1. SCF 함수 생성
   - SCF 콘솔이나 CLI를 통해 함수를 생성할 수 있습니다. 함수 생성을 위해 실행 환경 선택(함수 편집을 할 때 사용할 언어 선택), 함수 코드(온라인 편집이나 로컬 업로드 코드 패키지 지원) 제출이 필요합니다.
   - 미리 설정된 SCF 템플릿으로 생성 절차를 간소화시킬 수 있습니다. 자세한 내용은 함수 생성을 참조하십시오. 함수 작성 방법은 프로그래밍 언어에 따라 다릅니다. 자세한 내용은 SCF 문서를 참조하십시오.

2. 함수 테스트
   함수 생성 완료 후 테스트 템플릿 기능을 사용해 테스트를 진행할 수 있습니다. 테스트 템플릿은 COS 이벤트를 시뮬레이션하며 함수 실행을 트리거 합니다. 자세한 내용은 함수 테스트를 참조하십시오.
3. 트리거 추가
   테스트 완료 후 COS 트리거 생성을 통해 SCF 함수와 버킷을 바인딩할 수 있으며 콘솔이나 명령 라인으로 트리거를 추가할 수 있습니다. 자세한 내용은 트리거 생성 문서를 참조하십시오.
4. 실제 인증
   위 절차를 완료한 후 COS 버킷을 조작하여 전체 프로세스가 정상인지 인증할 수 있습니다. 예를 들어 콘솔, COS Browser 등의 툴을 통해 파일을 업로드 및 삭제하며 **SCF 콘솔** > **함수 세부 사항** > **일치하는 함수 명** > **실행 로그** 순서로 정상 작동하는지 인증할 수 있습니다.

SCF COS 트리거에 대한 자세한 내용은 COS 트리거 문서를 참조하십시오.

데이터 인덱스
Select 개요

최종 업데이트 날짜: : 2022-05-05 14:52:09

COS Select 기능은 구조화된 쿼리 명령(SQL)을 통해 Cloud Object Storage(COS)에 저장된 객체를 선별하여 객체나 사용자가 필요한 데이터를 인덱스하기 쉽도록 합니다. COS Select 기능으로 객체 데이터를 선별하여 COS에 전송된 데이터 양을 줄이면 데이터 검색에 필요한 비용과 딜레이 현상을 줄일 수 있습니다.

현재 COS Select 기능은 CSV, JSON, Parquet 포맷 형태의 버킷 객체와 GZIP이나 BZIP2로 압축된 객체(CSV, JSON 포맷의 객체) 인덱스를 지원합니다. 또한 COS Select 기능으로 결과물을 CSV나 JSON 포맷으로 지정할 수 있으며 결과 기록을 어떻게 구분할지 지정할 수 있습니다.

요청을 통해 SQL 표현식을 COS에 전송할 수 있습니다. 현재 COS Select는 일부 SQL 표현식만 지원합니다. COS Select가 지원하는 SQL 표현식에 대한 자세한 정보는 SQL 함수 문서를 참고하십시오.

COS 콘솔, API, SDK, COSCMD 등의 방식으로 SQL쿼리를 실행할 수 있습니다. COS 콘솔을 사용하여 파일 검색을 진행할 때는 일부 제한 사항이 있음을 주의하십시오. 최대 128M 파일 검색을 지원하며 데이터양은 40MB까지 반환됩니다. 더 많은 데이터를 인덱스 하려면 다른 방식을 사용해 주십시오.

> 설명 :
>
> - COS Select가 지원하는 데이터 유형과 현재 예약된 필드는 데이터 유형과 보관 필드에서 자세히 볼 수 있습니다.
> - 현재 인덱스 기능은 중국대륙 공유 클라우드 리전에서만 지원됩니다. 기타 리전에서는 이 기능을 지원하지 않습니다.

사용 제한

COS Select을 사용할 때 다음과 같은 제한 사항이 있습니다.

- 조회 대상의 `cos:GetObject` 권한을 가지고 있어야 합니다. 루트 계정은 기본적으로 이 권한을 가지고 있습니다.
- 표준 스토리지 유형의 객체만 인덱스할 수 있습니다.
- SQL 표현식의 최대 길이는 256KB입니다.
- 인덱스 결과에서 단일 기록의 최대 길이는 1MB입니다.

COS Select 기능이 지원하는 SQL 절은 다음과 같습니다.

- SELECT 명령
- FROM 절
- WHERE 절
- LIMIT 절

> 설명 :
> SQL 절에 대한 세부 정보는 Select 명령어를 참고하십시오.

현재 COS Select가 지원하는 함수는 다음 예시와 같습니다.

- 집계 함수: AVG 함수, COUNT 함수, MAX 함수, MIN 함수, SUM 함수
- 조건부 함수: COALESCE 함수, NULLIF 함수
- 변환 함수: CAST 함수
- 날짜 함수: DATE_ADD 함수, DATE_DIFF 함수, EXTRACT 함수, TO_STRING 함수, TO_TIMESTAMP 함수, UTCNOW 함수
- 문자열 함수: CHAR_LENGTH 함수, CHARACTER_LENGTH 함수, LOWER 함수, SUBSTRING 함수, TRIM 함수, UPPER 함수

> 설명 :
> SQL 함수에 대한 세부 정보는 SQL 함수를 참고하십시오.

현재 COS Select가 지원하는 오퍼레이터는 다음과 같습니다.

- 논리 오퍼레이터: `AND, NOT, OR`
- 비교 오퍼레이터: `&lt;, >, &lt;=, >=, =, &lt;>, !=, BETWEEN, IN`
- 패턴 매칭 오퍼레이터: `LIKE`
- 수학 오퍼레이터: `+, -, *, %`

> 설명 :
> 오퍼레이터에 관한 자세한 정보는 오퍼레이터를 참고하십시오.

인덱스 요청 보내기

콘솔, API, SDK등 여러 방식을 사용해 인덱스 요청을 할 수 있습니다.

- 콘솔을 사용하려면 인덱스 데이터 문서를 참고하여 작업할 수 있습니다.
- API를 사용하려면 SELECT Object Content API 문서를 참고하십시오.
- SDK를 사용하려면 SDK 개요에서 필요한 SDK 인터페이스를 선택하십시오.

FAQ

조회할 때 오류가 생기면 COS Select는 오류 코드와 관련 오류 메시지를 반환합니다. 관련 오류 코드와 설명 리스트는 Special error codes를 참고하십시오.

Select 명령어

최종 업데이트 날짜: : 2021-03-02 14:14:36

개요

COS Select 기능은 필요한 일부 데이터를 쉽게 인덱스할 수 있도록 SELECT SQL쿼리 명령을 지원하여 비용을 감소시키고 요청 딜레이를 줄일 수 있습니다. 다음은 SELECT 조회에서 지원하는 표준 절입니다.

- SELECT 명령
- WHERE 절
- LIMIT 절

> ⚠ **주의 :**
>
> 현재 COS Select는 절 쿼리 혹은 joins를 지원하지 않습니다.

SELECT 명령

SELECT 명령으로 COS 객체에서 보고 싶은 데이터를 인덱스할 수 있습니다. 이름, 함수, 표현식 등으로 데이터 조회가 가능하며 리스트의 형식으로 조회 결과를 반환합니다. SELECT 명령 포맷은 다음과 같습니다.

```
SELECT *
SELECT projection [ AS column_alias | column_alias ] [, ...]
```

첫번째 절의 SELECT 명령은 `*` (별표)가 표시되며 COS 객체에서 모든 열을 반환합니다. 두번째 절의 SELECT 명령은 사용자 지정을 사용하여 스칼라 표현식을 출력하고 **projection**은 모든 열에 사용자 지정 이름의 출력 리스트를 생성합니다.

WHERE 절

WHERE 절은 다음 구문을 사용합니다.

```
WHERE condition
```

WHERE 절은 **condition**을 통해 필터링을 진행합니다. **condition**은 Boolean 결과를 반환할 수 있는 표현식입니다. 반환 값이 TRUE인 행이어야 결과를 출력할 수 있습니다.

LIMIT 절

LIMIT 절은 다음 구문을 사용합니다.

```
LIMIT number
```

LIMIT 절은 조회 당 반환 되는 기록 수를 제한합니다. **number** 매개변수를 이용해 이 제한을 지정할 수 있습니다.

액세스 속성

SELECT와 WHERE 절은 파일 포맷이 CSV인지 JSON인지에 따라 다음 방식으로 조회할 필드를 선택할 수 있습니다.

CSV

- **열 번호**: 제 N열의 조회할 데이터는 `_N` 을 통해 지정할 수 있습니다. 어느 CSV 파일이든 열 번호는 1부터 증가합니다. 첫 열의 일련 번호가 `_1` 인 경우 두번째 열의 일련 번호는 `_2` 입니다. SELECT와 WHERE 절에서 `_N` 이나 `alias._N` 을 통해 조회할 열을 지정하는 것은 유효합니다.
- **열 헤더**: 조회할 CSV 파일에 헤더가 있다면 SELECT나 WHERE 절에서 이 헤더를 통해 조회가 필요한 열을 지정할 수 있습니다. SQL 명령에서는 SELECT나 WHERE 절의 `alias.column_name` 혹은 `column_name` 으로 지정합니다.

JSON

- **문서(Document)**: `alias.name` 방식을 사용하여 JSON 문서에 액세스할 수 있습니다. 중첩된 배열은 `alias.name1.name2.name3` 과 같은 방식으로 액세스할 수 있습니다.
- **리스트(List)**: 0부터 일련번호가 매겨지고 `[]` 오퍼레이터를 사용하는 인덱스 기능으로 리스트 요소에 액세스할 수 있습니다. 예시로 `alias[1]` 를 통해 JSON 리스트의 2번째 요소에 액세스할 수 있습니다. 중첩 배열에 액세스 해야 하는 경우 `alias.name1.name2[1].name3` 과 같은 방식으로 액세스할 수 있습니다.

예시

다음은 예시 데이터 샘플입니다.

```
{
"name": "Leon",
"org": "Tencent",
"projects":
[
{"project_name":"project1", "completed":true},
{"project_name":"project2", "completed":false}
```

```
]
}
```

- 예시1: 다음은 데이터 예시 중 이름을 조회하는 SQL 명령 및 조회 결과입니다.

```
Select s.name from COSObject s
```

```
{"name":"Leon"}
```

- 예시2: 다음은 데이터 예시 중 프로젝트 이름을 조회하는 SQL 명령 및 조회 결과입니다.

```
Select s.projects[0].project_name from COSObject s
```

```
{"project_name":"project1"}
```

헤더와 속성 이름의 대소문자 민감성

큰따옴표를 사용하여 CSV 파일의 헤더와 JSON 파일의 속성 이름을 표시하여 대소문자를 구분하는지 알 수 있습니다. 큰따옴표를 추가하지 않으면 헤더/속성 이름은 대소문자를 구분하지 않으며 명확하게 설정이 되어 있지 않은 경우 COS Select가 이상 경고를 보냅니다.

- 예시 1: 헤더/속성 이름에서 "NAME"이 있는 객체를 조회합니다.
  다음 SQL 예시는 큰따옴표를 사용하지 않았으므로 대소문자를 구분하지 않습니다. 이 헤더가 테이블에 포함되기 때문에 값이 반환될 수 있습니다.

```
SELECT s.name from COSObject s
```

  다음 SQL 예시는 큰따옴표를 사용하여 대소문자를 구분합니다. 테이블에서 이 헤더를 포함하지 않아 400 오류인 `SQLParsingError` 이 반환됩니다.

```
SELECT s."name" from COSObject s
```

- 예시2: 헤더/속성 이름을 조회할 때 "NAME"과 "name"객체가 있습니다.
  다음 SQL 예시는 큰따옴표를 사용하지 않아 대소문자를 구분하지 않습니다. 테이블에 "NAME"과 "name" 두 개의 헤더를 함께 포함하여 조회 명령 설정이 명확하지 않으므로 AmbiguousFieldName 이상 경고가 반환될 수 있습니다.

```
SELECT s.name from COSObject s
```

다음 SQL 예시는 큰따옴표를 사용하여 대소문자를 구분합니다. 테이블에 "NAME" 헤더가 포함되기 때문에 조회 결과가 반환될 수 있습니다.

```
SELECT s."NAME" from COSObject s
```

예약된 필드를 사용자 지정 필드로 사용하기

COS Select의 SQL 표현식에는 함수 이름, 데이터 유형, 오퍼레이터 등의 예약된 필드가 있습니다. 사용자는 예약된 필드를 CSV 파일 열 헤더 혹은 JSON 파일의 속성 이름으로 사용할 수 있는데 이때 예약된 필드와 충돌할 가능성이 있습니다. 이런 경우 큰따옴표를 사용하여 사용자 지정 필드가 사용 중임을 나타냅니다. 그렇지 않은 경우 COS가 `400 parse error` 오류를 반환할 수 있습니다.

완전한 예약된 필드 리스트를 조회해야 할 경우 예약된 필드를 참조하십시오.

- 예시: 조회할 객체의 헤더/속성 이름에는 예약된 필드 "CAST"가 포함되어 있습니다.
  다음 SQL 예시는 큰따옴표를 사용하여 CAST가 사용자 지정 필드임을 나타내므로 조회 결과가 성공적으로 반환될 수 있습니다.

  ```
  SELECT s."CAST" from COSObject s
  ```

  아래 SQL 예시는 CAST가 사용자 지정 필드임을 나타내기 위해 큰따옴표를 쓰지 않았으므로 COS는 예약된 필드로 처리를 하고 `400 parse error` 로 반환합니다.

  ```
  SELECT s.CAST from COSObject s
  ```

스칼라 표현식

SELECT 명령과 WHERE 절에서 SQL 스칼라 표현식(스칼라 표현식 반환)을 사용할 수 있습니다. 현재 COS Select는 다음과 같은 형식을 지원합니다.

- **literal**: SQL 텍스트.
- **column_reference**: column_name이나 alias.column_name.
- **unary_op expression**: SQL 단항 오퍼레이터.
- **expression binary_op expression**: SQL 이항 오퍼레이터.
- **func_name**: 호출된 스칼라 함수 이름.
- **expression** [NOT] BETWEEN **expression** AND **expression**
- **expression** LIKE **expression** [ESCAPE **expression**]

SQL 함수

최종 업데이트 날짜: : 2022-03-09 16:31:49

집계 함수

COS Select는 다음 집계 함수를 지원합니다.

함수 이름	매개변수 유형	반환 유형
AVG(expression)	INT,FLOAT,DECIMAL	입력 매개변수가 정수 형식일 경우 DECIMAL을 반환하며 부동 소수점 형식일 경우 FLOAT를 반환합니다. 그 외의 상황에서는 반환 값과 입력 매개변수가 일치합니다.
COUNT	-	INT
MAX(expression)	INT,DECIMAL	반환 값은 입력 매개변수와 일치합니다.
MIN(expression)	INT, DECIMAL	반환 값은 입력 매개변수와 일치합니다.
SUM(expression)	INT, FLOAT, DOUBLE, DECIMAL	입력 매개변수가 정수 형식일 경우 INT를 반환하며 부동 소수점 형식일 경우 FLOAT를 반환합니다. 그 외의 상황에서는 반환 값과 입력 매개변수가 일치합니다.

조건부 함수

COS Select는 다음 조건부 함수를 지원합니다.

COALESCE

COALESCE 함수는 순서대로 입력 매개변수를 결정하며 null이 아닌 최초의 매개변수 값을 반환합니다. 입력한 매개변수에 null이 아닌 매개변수가 포함되지 않는 경우 함수는 null 값을 반환합니다.

구문

```
COALESCE ( expression, expression, ... )
```

설명 :
expression 매개변수는 INT、String、Float 유형의 값, 배열, 중첩 배열 입력을 지원합니다.

예시

```
COALESCE(1) -- 1
COALESCE(1, null) -- 1
COALESCE(null, null, 1) -- 1
COALESCE(missing, 1) -- 1
COALESCE(null, 'string') -- 'string'
COALESCE(null) -- null
COALESCE(null, null) -- null
COALESCE(missing) -- null
COALESCE(missing, missing) -- null
```

NULLIF

NULLIF 함수는 두 개의 입력 매개변수 간의 차이를 판단하는데 두 개의 입력 매개변수 값이 같으면 NULL을 반환하고 그렇지 않으면 첫번째 입력 매개변수 값을 반환합니다.

구문

```
NULLIF ( expression1, expression2 )
```

> 설명 :
> expression 매개변수는 INT、String、Float 유형의 값, 배열, 중첩 배열 입력을 지원합니다.

예시

```
NULLIF(1, 2) -- 1
NULLIF(1, '1') -- 1
NULLIF(1, NULL) -- 1
NULLIF(1, 1) -- null
NULLIF(1.0, 1) -- null
NULLIF(missing, null) -- null
NULLIF(missing, missing) -- null
NULLIF([1], [1]) -- null
NULLIF(NULL, 1) -- null
NULLIF(null, null) -- null
```

변환 함수

COS Select는 다음 변환 함수를 지원합니다.

CAST

CAST 함수로 하나의 인스턴스를 다른 인스턴스로 변환할 수 있습니다. 인스턴스는 값일 수도 있고 어떤 확정 값을 계산할 수 있는 함수일 수도 있습니다.

구문

```
CAST ( expression AS data_type )
```

설명 :

- expression 매개변수는 값, 배열, 오퍼레이터 혹은 어떤 확정 값을 계산할 수 있는 SQL 함수일 수 있습니다.
- data type 매개변수는 변환 후의 데이터 유형이며 INT 유형을 예시로 들 수 있습니다. 현재 COS Select에서 지원하는 데이터 유형은 데이터 유형을 참고하십시오.

예시

```
CAST('2007-04-05T14:30Z' AS TIMESTAMP)
CAST(0.456 AS FLOAT)
```

날짜 함수

COS Select는 다음 날짜 함수를 지원합니다.

DATE_ADD

DATE_ADD 함수는 지정 타임스탬프(년, 월, 일, 시, 분, 초)에서 지정 시간만큼 더할 수 있으며 새로운 타임스탬프를 반환합니다.

구문

```
DATE_ADD( date_part, quantity, timestamp )
```

설명 :

- date_part 매개변수는 지정 타임스탬프에서 수정할 부분입니다. 년, 월, 일, 시, 분, 초가 될 수 있습니다.
- quantity 매개변수는 추가할 값을 나타내며 반드시 자연수여야 합니다.
- timestamp 매개변수는 수정할 타임스탬프입니다.

예시

```
DATE_ADD(year, 5, `2010-01-01T`) -- 2015-01-01
DATE_ADD(month, 1, `2010T`) -- 2010-02T
DATE_ADD(month, 13, `2010T`) -- 2011-02T
DATE_ADD(hour, 1, `2017T`) -- 2017-01-01T01:00-00:00
DATE_ADD(hour, 1, `2017-01-02T03:04Z`) -- 2017-01-02T04:04Z
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z
```

DATE_DIFF

DATE_DIFF 함수는 두 타임스탬프를 비교하고 두 타임스탬프의 차이 값을 반환합니다. 차이 값은 지정한 시간 단위를 보여줍니다. timestamp1의 date_part 값이 timestamp2보다 큰 경우 반환 값은 정수입니다. 그 반대의 경우는 음수를 반환합니다.

구문

```
DATE_DIFF( date_part, timestamp1, timestamp2 )
```

설명：

- date_part 매개변수는 두 타임스탬프를 비교한 시간 단위를 지정합니다. 년, 월, 일, 시, 분, 초일 수 있습니다.
- timestamp1 매개변수는 입력한 첫번째 타임스탬프입니다.
- timestamp2 매개변수는 입력한 두번째 타임스탬프입니다.

예시

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`) -- 1
DATE_DIFF(year, `2010T`, `2010-05T`) -- 4
DATE_DIFF(month, `2010T`, `2011T`) -- 12
DATE_DIFF(month, `2011T`, `2010T`) -- -12
DATE_DIFF(day, `2010-01-01T23:00T`, `2010-01-02T01:00T`) -- 0
```

EXTRACT

EXTRACT 함수는 지정된 타임스탬프에서 추출한 지정 시간 단위 값입니다.

구문

```
EXTRACT( date_part FROM timestamp )
```

설명：

- date_part 매개변수는 추출할 시간 단위를 지정합니다. 년, 월, 일 시, 분, 초일 수 있습니다.
- timestamp 매개변수는 입력한 타임스탬프입니다.

예시

```
EXTRACT(YEAR FROM `2010-01-01T`) -- 2010
EXTRACT(MONTH FROM `2010T`) -- 1
EXTRACT(MONTH FROM `2010-10T`) -- 10
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`) -- 3
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`) -- 4
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`) -- 7
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`) -- 8
```

TO_STRING

TO_STRING 함수로 타임스탬프를 지정 포맷 시간 문자열로 변환할 수 있습니다.

구문

```
TO_STRING ( timestamp time_format_pattern )
```

설명：

- timestamp 매개변수는 변환할 타임스탬프를 지정합니다.
- time_format_pattern 매개변수는 변환 시간 포맷을 지정합니다.

포맷	설명	예시
yy	연도 단위 2자리 수	98
y	연도 단위 4자리 수	1998
yyyy	연도 단위 고정 4자리 수, 부족한 부분은 0으로 보충	0199
M	월 단위	1

포맷	설명	예시
MM	월 단위 고정 2자리 수, 부족한 부분은 0으로 보충	01
MMM	월 단위 영어 약어	Jan
MMMM	월 단위 영어 전체 표기	January
MMMMM	월 단위의 첫 글자 약어	J(to_timestamp 함수에는 적합하지 않음)
d	한 달 중에 하루(1~31)	1
dd	일 수 중 고정 2자리 수(1~31)	01
a	오전 혹은 오후 표기(AM/PM)	AM
h	시간, 12시간제	1
hh	시각 고정 2자리 수이며 12시간제	01
H	시간, 24시간제	1
HH	시각 고정 2자리 수이며 24시간제	01
m	분(00-59)	1
mm	시간 고정 2자리 수이며 24시간제	01
s	분(00-59)	1
ss	시각 고정 2자리 수이며 24시간제	01
S	초의 소수 부분(정확도 0.1, 범위 0.0 - 0.9)	0
SS	초의 소수 부분(정확도 0.01, 범위 0.00 - 0.99)	6
SSS	초의 소수 부분(정확도 0.001, 범위 0.000 - 0.999)	60
...	...	...
SSSSSSSSS	초의 소수 부분(정확도 0.000000001, 범위 0.000000000 - 0.999999999)	60000000
n	나노 초	60000000
X	시간 단위 오프셋, 오프셋이 0이면 "Z"로 표시	+01 혹은 Z

포맷	설명	예시
XX 혹은 XXXX	시간 혹은 분 단위 오프셋, 오프셋이 0이면 "Z"로 표시	+0100 혹은 Z
xxx 혹은 xxxxx	시간 혹은 분 단위 오프셋, 오프셋이 0이면 "Z"로 표시	+01:00 혹은 Z
x	시간 단위 오프셋	1
xx 혹은 xxxx	시간 혹은 분 단위 오프셋	0100
xxx 혹은 xxxxx	시간 혹은 분 단위 오프셋	01:00

예시

```
TO_STRING(`1998-07-20T20:18Z`, 'MMMM d, y') -- "July 20, 1998"
TO_STRING(`1998-07-20T20:18Z`, 'MMM d, yyyy') -- "Jul 20, 1998"
TO_STRING(`1998-07-20T20:18Z`, 'M-d-yy') -- "7-20-69"
TO_STRING(`1998-07-20T20:18Z`, 'MM-d-y') -- "07-20-1998"
TO_STRING(`1998-07-20T20:18Z`, 'MMMM d, y h:m a') -- "July 20, 1998 8:18 PM"
TO_STRING(`1998-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX') -- "1998-07-20T20:18:00Z"
TO_STRING(`1998-07-20T20:18+08:00Z`, 'y-MM-dd''T''H:m:ssX') -- "1998-07-20T20:18:
00Z"
TO_STRING(`1998-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX') -- "1998-07-20T20:1
8:00+0800"
TO_STRING(`1998-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXXX') -- "1998-07-20T20:
18:00+08:00"
```

TO_TIMESTAMP

TO_TIMESTAMP 함수로 시간 문자열을 타임스탬프로 변환할 수 있습니다.

구문

```
TO_TIMESTAMP ( string )
```

설명：
string 매개변수는 입력한 시간 문자열입니다.

예시

```
TO_TIMESTAMP('2007T') -- `2007T`
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
```

UTCNOW

UTCNOW로 UTC 타임스탬프를 반환할 수 있습니다.

구문

```
UTCNOW()
```

예시

```
UTCNOW() -- 2019-01-01T14:23:12.123Z
```

문자열 함수

COS Select는 다음 문자열 함수를 지원합니다.

CHAR_LENGTH, CHARACTER_LENGTH

CHAR_LENGTH과 CHARACTER_LENGTH는 문자열의 문자 수를 계산해줍니다. 두 함수의 semantics는 같습니다.

구문

```
CHAR_LENGTH ( string )
```

> 설명：
> string 매개변수는 계산할 문자의 문자열을 지정합니다.

예시

```
CHAR_LENGTH('null') -- 4
CHAR_LENGTH('tencent') -- 7
```

LOWER

LOWER 함수는 문자열의 모든 대문자를 소문자로 변환할 수 있으며 대문자가 아닌 문자열은 수정할 수 없습니다.

구문

```
LOWER ( string )
```

설명：
string 매개변수는 대문자를 소문자로 변환해야 하는 문자열을 지정합니다.

예시

```
LOWER('TENcent') -- 'tencent'
```

SUBSTRING

SUBSTRING 함수는 문자열의 부속 문자열을 반환합니다. 하나의 인덱스를 지정하고 SUBSTRING 함수를 사용해 인덱스부터 부속 문자열 길이에 따라 기존 문자열의 나머지를 추출하여 결과를 반환합니다.

설명：
입력 문자열이 1개의 문자로 이루어졌고 인덱스 설정이 1보다 크면 SUBSTRING 함수가 그 문자열을 자동으로 1로 치환합니다.

구문

```
SUBSTRING( string FROM start [ FOR length ] )
```

설명：

- string 매개변수는 추출할 문자의 문자열을 지정합니다.
- start 매개변수는 문자열의 어떤 인덱스값이며 추출 위치를 지정합니다.
- length 매개변수는 문자 길이를 지정하며 문자 길이를 따로 지정하지 않은 경우 문자열의 나머지 부분이 추출됩니다.

예시

```
SUBSTRING("123456789", 0) -- "123456789"
SUBSTRING("123456789", 1) -- "123456789"
SUBSTRING("123456789", 2) -- "23456789"
```

```
SUBSTRING("123456789", -4) -- "123456789"
SUBSTRING("123456789", 0, 999) -- "123456789"
SUBSTRING("123456789", 1, 5) -- "12345"
```

TRIM

TRIM 함수는 지정 문자열의 첫 글자 앞 혹은 끝 글자 뒤의 모든 문자 부호를 삭제할 수 있습니다. 기본적으로 삭제하는 문자 부호는 ""입니다.

구문

```
TRIM ( [[LEADING | TRAILING | BOTH remove_chars] FROM] string )
```

설명 :

- string 매개변수는 작업에 필요한 문자열을 지정합니다.
- `LEADING | TRAILING | BOTH` 매개변수는 삭제할 문자열 앞(LEADING) 혹은 문자열 뒤(TRAILING) 혹은 두 곳 모두 삭제(BOTH)할 부분의 나머지 문자를 지정합니다.
- remove_chars 매개변수는 삭제할 나머지 문자열을 지정합니다. remove_chars 매개변수는 1개의 문자보다 길 수 있으며 TRIM 함수는 string 매개변수 앞뒤에서 나머지 문자열을 식별하여 삭제 처리합니다.

예시

```
TRIM(' foobar ') -- 'foobar'
TRIM(' \tfoobar\t ') -- '\tfoobar\t'
TRIM(LEADING FROM ' foobar ') -- 'foobar'
TRIM(TRAILING FROM ' foobar ') -- 'foobar'
TRIM(BOTH FROM ' foobar ') -- 'foobar'
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'
```

UPPER

UPPER 함수는 문자열의 모든 소문자를 대문자로 변환할 수 있으며 소문자가 아닌 문자열은 수정할 수 없습니다.

구문

```
UPPER ( string )
```

설명 :

string 매개변수는 변환할 대문자 문자열을 지정합니다.

예시

```
UPPER('tenCENT') -- 'TENCENT'
```

필드 보관

최종 업데이트 날짜: : 2021-03-02 14:13:49

COS Select는 기능의 정상 작동과 후속 확장을 보장하기 위해 함수 이름, 데이터 유형, 오퍼레이터 등의 예약된 필드가 있습니다. SQL 쿼리 명령을 사용할 때 이 필드를 사용하면 조회하는 데 도움을 받을 수 있습니다.

시리얼 넘버	필드	시리얼 넘버	필드	시리얼 넘버	필드	시리얼 넘버	필드
1	absolute	51	create	101	goto	151	octet_leng
2	action	52	cross	102	grant	152	of
3	add	53	current	103	group	153	on
4	all	54	current_date	104	having	154	only
5	allocate	55	current_time	105	hour	155	open
6	alter	56	current_timestamp	106	identity	156	option
7	and	57	current_user	107	immediate	157	or
8	any	58	cursor	108	in	158	order
9	are	59	date	109	indicator	159	outer
10	as	60	day	110	initially	160	output
11	asc	61	deallocate	111	inner	161	overlaps
12	assertion	62	dec	112	input	162	pad
13	at	63	decimal	113	insensitive	163	partial
14	authorization	64	declare	114	insert	164	pivot
15	avg	65	default	115	int	165	position
16	bag	66	deferrable	116	integer	166	precision
17	begin	67	deferred	117	intersect	167	prepare
18	between	68	delete	118	interval	168	preserve
19	bit	69	desc	119	into	169	primary

20	bit_length	70	describe	120	is	170	prior
21	blob	71	descriptor	121	isolation	171	privileges
22	bool	72	diagnostics	122	join	172	procedure
23	boolean	73	disconnect	123	key	173	public
24	both	74	distinct	124	language	174	read
25	by	75	domain	125	last	175	real
26	cascade	76	double	126	leading	176	reference
27	cascaded	77	drop	127	left	177	relative
28	case	78	else	128	level	178	restrict
29	cast	79	end	129	like	179	revoke
30	catalog	80	end-exec	130	limit	180	right
31	char	81	escape	131	list	181	rollback
32	char_length	82	except	132	local	182	rows
33	character	83	exception	133	lower	183	schema
34	character_length	84	exec	134	match	184	scroll
35	check	85	execute	135	max	185	second
36	clob	86	exists	136	min	186	section
37	close	87	external	137	minute	187	select
38	coalesce	88	extract	138	missing	188	session
39	collate	89	false	139	module	189	session_u
40	collation	90	fetch	140	month	190	set
41	column	91	first	141	names	191	sexp
42	commit	92	float	142	national	192	size
43	connect	93	for	143	natural	193	smallint
44	connection	94	foreign	144	nchar	194	some

45	constraint	95	found	145	next	195	space
46	constraints	96	from	146	no	196	sql
47	continue	97	full	147	not	197	sqlcode
48	convert	98	get	148	null	198	sqlerror
49	corresponding	99	global	149	nullif	199	sqlstate
50	count	100	go	150	numeric	200	string

데이터 유형

최종 업데이트 날짜: : 2021-03-02 14:15:10

개요

COS Select는 여러 가지 원시 자료형을 지원합니다.

설명 :

컴파일러가 직접 지원하는 데이터 유형을 **원시 자료형**이라고 합니다.

데이터 유형 변환

COS Select는 CAST 함수를 통해 입력한 데이터 유형을 확정합니다. 일반적으로 CAST 함수를 통해 데이터 유형을 지정하지 않은 경우 COS Select는 입력한 데이터 유형을 string 유형으로 인식합니다.

CAST 함수에 관한 정보를 더 자세히 알고 싶다면 SQL 함수 문서 중 CAST 부분을 참조하십시오.

지원하는 데이터 유형

COS Select는 다음 원시 자료형을 지원합니다.

이름	설명	예시
bool	TRUE/FALSE	FALSE
int, integer	8바이트는 부호 정수가 있습니다. 범위는 -9,223,372,036,854,775,808 - 9,223,372,036,854,775,807입니다.	100000
string	UTF-8코드의 문자열, 문자 길이 범위는 1 - 2,147,483,647입니다.	'xyz'
float	8바이트 부동 소수점	CAST(0.456 AS FLOAT)
decimal, numeric	10진법 값으로 최대 정확도는 소수점 38자리까지 나타나며 값의 범위는 -2^{31} - $2^{31}-1$입니다.	123.456
timestamp	타임스탬프는 특정 시각을 나타내며 여러가지 시각 표현을 지원합니다. 텍스트 포맷 타임스탬프는 W3C 규범을 따르지만 “T”로 끝나야 합니다(“일”단위 기록	CAST('2007-04-

	은 제외). 분수 초를 사용할 경우 적어도 소수점 1자리 수까지는 표시해야 하며 소수점 자리 수에 제약이 없습니다. 로컬 타임 오프셋은 UTC와 비교했을 때의 시와 분 오프셋으로 나타내거나 “Z”는 UTC와 비교했을 때의 로컬 시간 오프셋을 나타내기 위해 사용될 수 있습니다. 날짜를 기록할 경우에는 타임 오프셋을 표시할 필요가 없습니다.	05T14:30Z' AS TIMESTAMP)

오퍼레이터

최종 업데이트 날짜: : 2021-03-02 14:15:24

COS Select는 다음 오퍼레이터를 지원합니다.

오퍼레이터 종류	오퍼레이터
논리 오퍼레이터	AND, NOT, OR
비교 오퍼레이터	<, >, <=, >=, =, <>, !=, BETWEEN, IN. 예시 `IN ('a', 'b', 'c')`
패턴 매칭 오퍼레이터	LIKE
수학 오퍼레이터	+, -, *, %

오퍼레이터 우선 순위

다음 표에서는 내림차순으로 오퍼레이터의 작업 우선 순위를 보여줍니다.

오퍼레이터/요소	관련성	사용 시나리오
-	우	단항 뺄셈
*, /, %	좌	곱셈, 나눗셈, 모듈로
+, -	좌	덧셈, 뺄셈
IN	-	구성원 자격 설정
BETWEEN	-	() 범위 내
LIKE	-	문자열 패턴 매칭
<>	-	보다 작음, 보다 큼
=	우	같음, 할당
NOT	우	논리 NOT
AND	좌	논리 AND
OR	좌	논리 OR

로그 관리
로그 관리 개요

최종 업데이트 날짜: : 2023-03-14 17:05:32

소개

로그 관리 기능은 버킷을 더 편리하게 관리할 수 있도록 지정한 원본 버킷의 액세스 상세 정보를 기록하고 해당 정보를 로그 파일 포맷으로 지정할 수 있습니다.

대상 버킷에서 로그 기록 경로는 다음과 같습니다.

```
대상 버킷/경로 접두사{YYYY}/{MM}/{DD}/{time}_{random}_{index}
```

로그는 5분마다 생성되어 한 줄씩 기록됩니다. 모든 기록은 여러 개의 필드를 포함하고 각 필드는 빈칸으로 분할됩니다. 주의할 점은 단일 로그 파일의 최대 크기가 256MB이며, 5분 이내 발생된 로그량이 256MB을 초과할 경우 로그는 여러 개의 로그 파일로 분할된다는 것입니다. 현재 지원하는 로그 필드는 다음과 같습니다.

필드 시리얼 넘버	이름	의미	예시
1	eventVersion	기록 버전	1.0
2	bucketName	버킷 이름	examplebucket-1250000000
3	qcsRegion	요청 리전	ap-beijing
4	eventTime	이벤트 시간(요청 종료 시간, UTC 0시 타임스탬프)	2018-12-01T11:02:33Z
5	eventSource	사용자의 액세스 도메인	examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
6	eventName	이벤트 이름	UploadPart
7	remoteIp	출처 IP	192.168.0.1
8	userSecretKeyId	사용자 액세스 KeyId	AKIDNYVCdoJQyGJ5b1234

필드 시리얼 넘버	이름	의미	예시
9	reservedFiled	예약된 필드	예약된 필드는 `-` 로 나타납니다.
10	reqBytesSent	요청 바이트 수(Bytes)	83886080
11	deltaDataSize	스토리지 양 변경 요청 (Bytes)	808
12	reqPath	요청한 파일 경로	/folder/text.txt
13	reqMethod	요청 방법	put
14	userAgent	사용자 UA	cos-go-sdk-v5.2.9
15	resHttpCode	HTTP 반환 코드	404
16	resErrorCode	에러 코드	NoSuchKey
17	resErrorMsg	에러 정보	The specified key does not exist.
18	resBytesSent	반환 바이트 수(Bytes)	197
19	resTotalTime	요청 총 소요 시간(밀리 초, 마지막 바이트 응답 시간-첫 바이트 요청 시간에 해당)	4295
20	logSourceType	로그 소스 유형	USER(사용자 액세스 요청), CDN(CDN 원본 요청)
21	storageClass	스토리지 유형	STANDARD, STANDARD_IA, ARCHIVE
22	accountId	버킷 소유자 ID	100000000001

필드 시리얼 넘버	이름	의미	예시
23	resTurnAroundTime	요청 서버 소요 시간(밀리초, 첫 바이트 응답 시간-마지막 바이트 요청 시간에 해당)	4295
24	requester	요청자 계정. 요청자가 루트 계정인 경우 이 매개변수는 `루트 계정 UIN:루트 계정 UIN` 형식입니다. 요청자가 서브 계정인 경우 이 매개변수는 `루트 계정 UIN:서브 계정 UIN` 형식입니다. 요청자가 Service Role 인 경우 이 매개변수는 `승인된 루트 계정 UIN:역할 ID` 형식입니다. 요청자가 익명인 경우 이 매개변수는 `-` 로 표시됩니다	100000000001:100000000001
25	requestId	요청 ID	NWQ1ZjY4MTBfMjZiMjU4NjRfOWI1N180NDBiYTY=

필드 시리얼 넘버	이름	의미	예시
26	objectSize	객체 크기 (Bytes)	808, 멀티파트 업로드를 사용할 경우 objectSize 필드는 업로드를 완료했을 때만 나타나며, 각 멀티파트 업로드 기간에 해당 필드는 `-` 로 나타남
27	versionId	객체 버전 ID	랜덤 문자열
28	targetStorageClass	대상 스토리지 유형, 복사 작업 요청을 시작하면 해당 필드를 기록함	STANDARD, STANDARD_IA, ARCHIVE
29	referer	요청 HTTP referer	`*.example.com` 또는 111.111.111.1
30	requestUri	요청 URI	"GET /fdgfdgsf%20/%E6%B5%AE%E7%82%B9%E6%95%B0 HTTP/1.1"
31	vpcId	VPC 요청 ID	"0"(VPC 아님)/"12345"(VPC, "0"이 아닌string)

주의 :

- 현재 COS의 로그 관리 기능이 지원되는 리전은 베이징, 상하이, 광저우, 난징, 충칭, 청두, 중국홍콩, 서울, 싱가포르, 토론토, 실리콘밸리, 뭄바이입니다.
- 로그 관리 기능은 소스 버킷과 대상 버킷이 반드시 같은 리전에 있어야 합니다.
- 로그를 보관한 대상 버킷은 소스 버킷 자체가 될 수 있지만, 권장하지는 않습니다.
- 현재 XML API 및 XML API 기반으로 구현된 SDK, 툴 등에서 버킷 액세스를 요청했을 경우에만 로그가 기록됩니다. JSON API 및 JSON API 기반으로 구현된 SDK, 툴 등에서의 액세스는 로그를 기록하지 않습니다.
- 사용자의 필요 및 업무 진행 상황에 따라 COS는 액세스 로그 중 필드를 추가할 수 있으므로 로그 리졸브 시 해당 프로세스를 진행하십시오.

로그 관리 활성화

콘솔 사용

사용자는 콘솔을 통해 로그 관리 기능을 빠르게 활성화할 수 있습니다. 작업 가이드는 로그 관리 설정 콘솔 가이드를 참고하십시오.

API 사용

API를 사용해 지정 버킷에 로그 관리 기능을 활성화할 경우 다음 순서를 참고하십시오.

1. 로그 역할을 생성합니다.
2. 로그 역할로 권한을 바인딩합니다.
3. 로그 관리를 활성화합니다.

1. 로그 역할 생성

로그 역할 생성과 구체적인 인터페이스 정보는 CreateRole을 참고하십시오.
roleName은 반드시 CLS_QcsRole이어야 합니다.
policyDocument:

```
{
"version": "2.0",
"statement": [{
"action": "name/sts:AssumeRole",
"effect": "allow",
"principal":{
"service": "cls.cloud.tencent.com"
}
}]
}
```

2. 로그 역할 바인딩 권한

역할 권한 바인딩 권한과 구체적인 인터페이스 정보는 AttachRolePolicy를 참고하십시오.
policyName은 QcloudCOSAccessForCLSRole, roleName은 1단계의 CLS_QcsRole 또는 roleName 생성 시 반환된 roleID를 사용할 수 있습니다.

3. 로그 관리 활성화

인터페이스 호출 및 로그 관리 기능 활성화와 구체적인 인터페이스 정보는 PUT Bucket logging을 참고하십시오. 로그를 보관한 대상 버킷과 소스 버킷은 같은 리전에 있어야 합니다.

로그 관리 제한

최종 업데이트 날짜: : 2021-03-02 11:38:50

로그 관리 기능은 지정된 소스 버킷에 대한 자세한 액세스 정보를 기록하고, 이 정보를 로그 파일 형태로 지정된 버킷에 저장하여 버킷을 보다 더 잘 관리할 수 있습니다.

현재 액세스 로그 관리 기능의 사용 제한은 다음과 같습니다.

- 전달 빈도 제한: 로그는 5분마다 생성됩니다.
- 전달 로그 파일 크기 제한: 매번 전달되는 로그 파일의 최대 크기는 256MB로, 해당 제한을 초과할 시 새로운 파일로 전달됩니다.
- 전달 로그 포맷: 한 줄씩 기록되며, 각각의 기록은 여러 개의 필드를 포함하고 필드 간에는 빈칸으로 분할됩니다.
- 전달 필드 제한: 전달 필드 관련 내용은 로그 관리 개요의 설명을 참조하십시오.
- 무효 필드 설명: 로그 중 `-` 이 문자가 존재한다면, 해당 필드가 무효 또는 기본값으로 기록됐다는 의미입니다.

로그에 기록되는 내용

- 사용자가 시작한 업로드/다운로드/객체 삭제, 및 버킷 생성/삭제, 버킷 설정 수정 등의 요청을 기록합니다.
- 사용자가 CDN을 통해 콘텐츠를 배포하고, CDN에서 COS 원본 서버로 데이터를 가져올 때 발생한 요청을 기록합니다.

로그에 기록되지 않는 내용

- 오프라인 원본 요청: 사용자가 원본을 설정한 후, COS에 객체가 없으면 사용자가 지정한 원본 서버에서 데이터를 다운로드합니다. 이 다운로드 작업, 즉 오프라인 원본 요청은 로그에 기록되지 않습니다.
- 정적 웹 사이트 내부 리디렉션 작업: 사용자가 정적 웹 사이트 기능에서 리디렉션을 설정했을 경우, index.html에 액세스하면 다른 페이지로 리디렉션될 수 있으며, 이런 리디렉션 작업은 로그에 기록되지 않습니다.
- 라이프사이클 전환, 삭제 등 작업: 사용자가 라이프사이클 기능을 설정해 객체에 만료 전환 또는 삭제를 진행할 경우, 이런 전환 및 삭제 작업은 COS 백그라운드에서 이뤄지며 로그에 기록되지 않습니다.
- 객체 리스트 나열 및 업로드 리스트 보고 작업: 리스트 기능은 사용자를 대신해 주기적으로 버킷 내 전체 또는 지정 객체를 나열하고, 생성된 리스트 보고를 사용자 버킷으로 전달하지만, 객체 나열 및 리스트 보고 전달 작업은 로그에 기록되지 않습니다.
- 리전 간 복제 객체 작업: 리전 간 복제 기능은 소스 버킷에서 객체를 얻고 객체를 타깃 버킷으로 업로드해야 합니다. 이런 작업은 COS 백그라운드에서 이뤄지며 로그에 기록되지 않습니다.
- COS Select 기능 중 객체 다운로드 작업: COS Select 기능은 사용자의 객체 인덱스를 도와주며, 먼저 스토리지 디바이스에서 데이터를 가져와야 인덱스가 가능합니다. 객체 다운로드 작업은 COS 백그라운드에서 이뤄지며 로그에 기록되지 않습니다.

COS를 사용하여 Tencent Cloud 제품 로그 저장

최종 업데이트 날짜: : 2022-05-05 14:52:08

소개

Tencent Cloud 서비스를 사용할 때 대량의 로그가 생성되는데, 이 로그들은 귀하의 업무 상황을 기록해 업무 상황을 분석하는 데 도움이 되며 귀하의 업무 발전과 의사 결정에 도움을 줍니다. COS 스토리지 기능을 이용해 클라우드 서비스 로그를 지속적으로 저장할 수 있으며, API, SDK 또는 툴 등을 통해 COS로부터 로그를 쉽게 얻고 분석할 수 있습니다.

COS 스토리지 클라우드 서비스 로그를 사용하면 다음 문제를 해결할 수 있습니다.

- **영구 스토리지**: COS는 안정된 영구 스토리지 서비스를 제공하므로 낮은 비용으로 로그를 COS로 저장해 영속화 저장이 가능합니다. 업무 필요에 의해 로그를 기반으로 분석 또는 의사 결정 시, COS를 통해 언제 어디서나 임의 시간대의 로그를 얻을 수 있습니다.
- **데이터 인덱스**: COS는 Select 기능을 제공하므로 해당 기능으로 COS에 저장된 로그에 대한 간단한 인덱스 및 추출이 가능합니다. 로그의 필드를 결합하면 필요한 정보를 인덱스하고 데이터 다운로드 트래픽을 줄일 수 있습니다.
- **데이터 분석**: Sparkling 제품을 사용하여 COS에 저장된 로그를 분석할 수 있으며, 하나 이상의 로그 파일을 선택하고 Sparkling을 사용하여 로그를 분석하고 분석 결과를 기반으로 의사 결정을 내릴 수 있습니다.

로그 전송 방식

두 가지 방식으로 Tencent Cloud 서비스의 로그를 COS에 저장할 수 있습니다.

- 클라우드 서비스 자체 로그 전송 기능 사용: 예를 들어 COS, CA 등 제품은 로그를 COS로 직접 전송할 수 있습니다.
- 로그 서비스 CLS의 전송 기능 사용: CLS로 전송된 클라우드 서비스 로그는 CLS를 통해 COS로 전송돼 영구 저장됩니다.

현재 Tencent Cloud 서비스의 이 두 가지 방식에 대한 지원 상황은 다음과 같습니다.

클라우드 서비스 이름	COS로의 직접 전송 지원 여부	CLS로의 전송 지원 여부
CA	지원	미지원

클라우드 서비스 이름	COS로의 직접 전송 지원 여부	CLS로의 전송 지원 여부
CLB	미지원	지원
CKafka	지원	미지원
API Gateway	미지원	지원
SCF	미지원	지원
TKE	미지원	지원
CSS	미지원	지원
TCB	미지원	지원(단, CLS를 통한 COS로의 전송은 지원하지 않음)
COS	지원	얼로우리스트 활성화 신청을 지원합니다. 고객센터를 통해 얼로우리스트 활성화

COS로 직접 로그 전송하기

다음 Tencent Cloud 서비스는 COS로 직접 로그를 전송하는 기능을 갖고 있습니다. 해당 제품 문서 가이드에 따라 로그 전송 규칙을 설정하여 로그를 COS로 전송할 수 있습니다.

클라우드 서비스 이름	로그 전달 문서	로그 전달 간격	로그 전달 경로
CA	알아보기	10-15분	cloudaudit/customprefix/timestamp
CKafka	알아보기	5분-60분 전송 간격 지정 가능	instance id/topic id/timestamp
COS	알아보기	5분	경로 접두사는 자동 지정되며, 식별 가능한 경로 설정을 권장합니다. 예시: cos_bucketname_access_log/timestamp

설명：
메시지 큐는 해당 제품에서 생성된 정보 데이터의 전송을 지원합니다. CKafka 인스턴스 생성 등 동작 로그를 얻어야 할 경우, CA 제품을 전송할 로그를 선택하십시오.

CLS를 통해 COS로 로그 전송하기

다음 Tencent Cloud 서비스는 사용자가 인덱스 및 분석을 할 수 있도록 로그를 CLS로 전송하는 기능을 제공합니다. CLS는 COS로 전송하는 제품 기능도 동시에 제공하여, 사용자가 로그를 쉽게 영구 저장할 수 있게 합니다. CLS로의 로그 전송을 지원하는 제품도 CLS 밖에서 COS로 전송된 로그를 활성화하는 방식을 통해 데이터를 영구 저장해 스토리지 비용을 낮추고 보다 편리한 오프라인 분석이 가능합니다. 현재 CLS로의 직접 로그 전송을 지원하는 제품은 다음과 같습니다.

클라우드 서비스 이름	로그 전송 문서
API Gateway	알아보기
TKE	알아보기
LVB	알아보기

CLS에서 COS로의 전송은 다음 세 가지 방식으로 지원됩니다.

- 세퍼레이터 포맷 전송: 데이터를 세퍼레이터 포맷에 따라 COS로 전송합니다. 자세한 내용은 CSV Shipping을 참고하십시오.
- JSON 포맷 전송: 데이터를 JSON 포맷에 따라 COS로 전송합니다. 자세한 내용은 JSON Shipping을 참고하십시오.
- 원문 포맷 전송: 데이터를 원문 포맷에 따라 전송합니다. 단일 행 텍스트, 다중 행 텍스트 전송을 지원하며, 부분적으로 CSV 포맷 전송을 지원합니다. 자세한 내용은 Source Format Shipping을 참고하십시오.

CLS를 통해 COS로 로그를 전송하려면 다음 작업을 수행해야 합니다.

1. 업무 필요에 따라 해당 제품을 선택하고, 위에서 제공한 제품 로그 전송 문서 링크 가이드에 따라 로그셋 및 로그 테마를 설정하면, 업무에서 발생한 데이터가 CLS로 연결됩니다.
2. 이후, 업무 필요에 따라 적합한 포맷을 선택해 데이터를 COS로 전송합니다. 로그를 COS로 보낼 때, 제품 이름을 경로 접두사로 삼아 서로 다른 제품 로그를 구분하길 권장합니다. 예를 들어, TKE 로그는 `TKE_tkeid_log/timestamp` 로 이름을 생성합니다.
3. 전송 규칙 설정이 끝나면 별도로 SCF 제품 아래 파일 업로드 이벤트 공지를 설정할 수 있습니다. 로그 데이터가 COS로 전송되면 이벤트 공지에 따라 다음 단계 작업을 수행할 수 있습니다. 자세한 내용은 이벤트 알림을 참고하십시오.

로그 분석

로그를 로컬로 다운로드해 오프라인 분석하기

로그 데이터를 로컬로 다운로드해야 할 경우, 콘솔/SDK/API 또는 툴 등 다양한 방식으로 다운로드할 수 있습니다. 다음은 다운로드 방식에 대한 사용 문서 설명으로, 다음 문서를 참고해 코드에서 파일 경로와 관련된 부분을 로그 저장 경로로 바꾸면 로그 데이터를 로컬로 다운로드할 수 있습니다.

다운로드 방식	사용 설명
콘솔	알아보기
cosbrowser	알아보기
coscmd	알아보기
Android SDK	알아보기
C SDK	알아보기
C++ SDK	알아보기
.NET SDK	알아보기
Go SDK	알아보기
iOS SDK	알아보기
Java SDK	알아보기
JavaScript SDK	알아보기
Node.js SDK	알아보기
PHP SDK	알아보기
Python SDK	알아보기
미니프로그램 SDK	알아보기
API	알아보기

COS Select를 사용해 로그 분석하기

COS Select 기능을 통해 COS에 저장된 로그 파일을 직접 인덱스 및 분석할 수 있습니다. 전제 조건은 로그 파일을 CSV 또는 JSON 포맷으로 저장하는 것입니다. COS Select 기능을 통해 필요한 로그 필드를 선택할 수 있으며, 이로써 COS에서 전송하는 로그 데이터양을 크게 줄여 사용 비용을 줄임과 동시에 데이터 획득 효율을 높이게 됩니다. COS Select 기능에 대한 자세한 내용은 Select 개요를 참고하십시오.

현재 콘솔 또는 API 방식을 통해 COS Select 기능을 사용할 수 있습니다.

사용 방식	사용 설명
콘솔	알아보기
API	알아보기

데이터 재해 복구
버전 제어
버전 제어 개요

최종 업데이트 날짜: : 2022-05-20 11:36:23

소개

버전 제어는 동일한 버킷에 동일한 객체의 여러 버전을 저장하는 데 사용합니다. 예를 들어, 한 버킷에 객체 키가 picture.jpg로 동일하지만 버전 ID가 다른(예: 100000, 100101, 120002 등) 여러 객체를 저장할 수 있습니다. 사용자는 버킷에 버전 제어 기능을 활성화한 후 버전 ID에 따라 버킷에 있는 객체를 인덱스, 삭제하거나 복구할 수 있으며, 이는 사용자가 잘못 삭제하거나 응용 프로그램 장애로 손실된 데이터를 복구하는데 도움이 됩니다. 예를 들어, 버전 제어가 활성화된 객체를 삭제하는 경우 다음과 같습니다.

- 객체를 삭제할 경우(비 완전 삭제) COS는 삭제할 객체에 삭제 마커를 삽입하며 해당 마커가 현재 객체 버전이 됩니다. 사용자는 **삭제 마커**에 따라 이전 버전을 복구할 수 있습니다.
- 객체를 대체할 경우 COS는 새로 업로드하는 객체에 새로운 버전 ID를 삽입하며, 버전 ID에 따라 교체하기 이전의 객체로 복구할 수 있습니다.

버전 제어 상태

버킷에는 버전 제어 비활성화 상태, 버전 제어 활성화 상태, 버전 제어 일시 중지 상태의 세 가지 버전 제어 상태가 있습니다.

- **버전 제어 비활성화 상태:** 버킷의 기본 초기화 상태로, 버전 제어 기능이 비활성화되어 있는 상태입니다.
- **버전 제어 활성화 상태:** 버킷 버전 제어 기능이 활성화되어 있는 상태로, 해당 버킷의 **모든 객체**에 적용됩니다. 버킷에 처음 버전 제어를 활성화하면 해당 버킷에 새로 업로드하는 객체는 고유의 버전 ID를 갖게 됩니다.
- **버전 제어 일시 중지 상태:** 버킷에 버전 제어 기능을 활성화했다 일시 중지한 상태이며(버전 제어 비활성화 상태로 돌아갈 수 없음), 이후에 버킷에 업로드하는 객체는 버전 제어 객체로 저장되지 않습니다.

주의 :

1. 일단 버킷에 버전 제어를 활성화하면 버전 제어 비활성화 상태(초기 상태)로 돌아갈 수 없습니다. 단, 해당 버킷의 버전 제어를 일시 중지할 수 있으며, 일시 중지하는 경우 이후 업로드하는 객체는 여러 버전이 생성되지 않습니다.
2. 버전 제어 활성화 전 버킷에 저장된 객체의 버전 ID는 모두 null입니다.

3. 버전 제어 활성화 또는 일시 중지 시, COS에서 해당 객체를 처리하는 요청 방식이 변경되며 객체 자체는 변경되지 않습니다.
4. 루트 계정과 권한이 부여된 서브 계정으로만 버킷 버전 제어 기능을 일시 중지할 수 있습니다.

버전 제어 상태에서의 객체 관리

서로 다른 버전 제어 상태인 버킷에서 각 상태의 버킷에 저장되어 있는 객체를 업로드, 조회, 삭제할 수 있습니다. 버전 제어 비활성화 상태를 제외하고, 버전 제어 활성화 상태와 일시 중지 상태에서는 버킷에 저장되어 있는 객체를 조회할 수 있으며, 버전 ID 미지정 및 지정 객체를 삭제할 수 있습니다.

- 버전 제어 비활성화 상태: 객체의 업로드, 조회, 삭제 등 작업 방식은 변경되지 않으며, 자세한 내용은 객체 관리 목록의 문서를 참조하십시오.
- 버전 제어 활성화 및 일시 중지 상태: 객체의 업로드, 조회, 삭제 등 작업 방식에서 이전과 다른 점은 버전 ID가 삽입된다는 점이며, 또한 객체 삭제 작업 시 "삭제 마커" 개념이 추가됩니다.

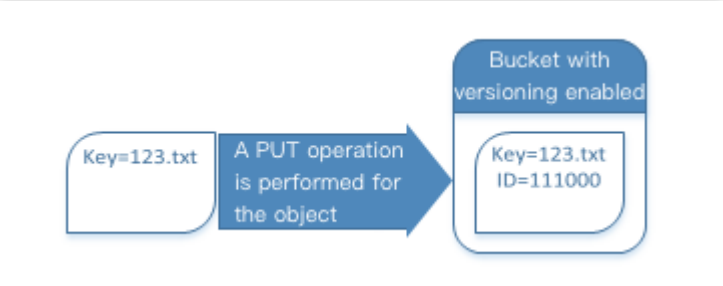
버전 제어 활성화 상태에서의 객체 관리

버킷 버전 제어 활성화 전 이미 버킷에 존재하는 객체의 보전 ID는 null입니다. 버전 제어 활성화 후 버킷에 이미 존재하는 객체는 변경되지 않으며, COS가 이미 존재하는 객체를 처리하는 방식만(예: 요청 방식) 변경됩니다. 이때 새로 업로드하는 동일한 이름의 객체는 다른 버전으로 동일한 버킷에 존재하게 됩니다. 다음과 같이 버전 제어를 활성화한 버킷에서의 객체 관리를 소개합니다.

주의 :
버전 제어를 비활성화한 버킷과 활성화한 버킷에 객체를 업로드하는 방법은 동일하지만, 해당 버전 ID는 동일하지 않습니다. 버전 제어를 활성화하는 경우 COS는 객체에 특정 버전 ID를 할당하며, 비활성화하는 경우 업로드하는 객체의 버전 ID는 항상 null이 됩니다.

객체 업로드

버킷에 버전 제어를 활성화한 후 사용자가 PUT, POST 또는 COPY 작업을 실행하면 COS는 자동으로 해당 버킷에 저장된 객체에 고유의 버전 ID를 추가합니다.
다음 이미지와 같이 버전 제어를 활성화한 버킷에 객체를 업로드할 경우 COS는 해당 객체에 고유의 버전 ID를 추가합니다.



버전 제어 객체 열거

COS는 버킷과 연결된 versions 매개변수에 객체 버전 정보를 저장합니다. 저장 시간 우선 순서에 따라 객체 버전을 반환하며, 가장 우선적으로 최근 저장된 버전을 반환합니다.

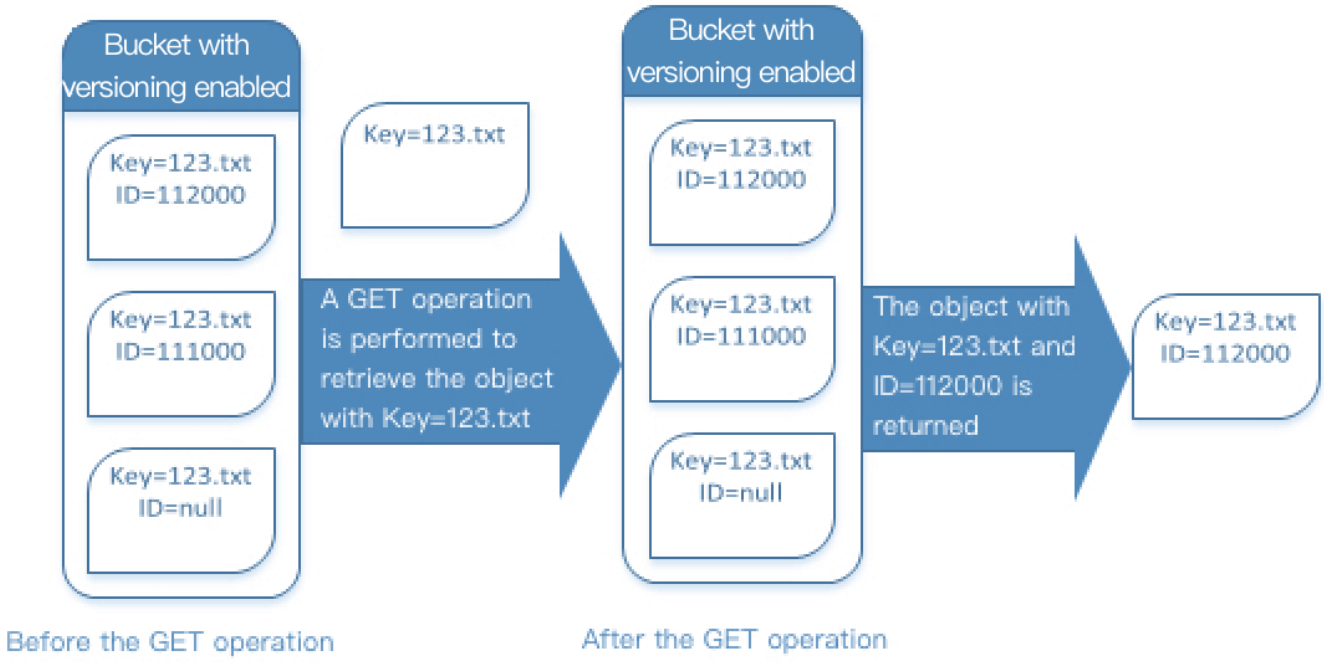
특정 객체의 모든 버전 조회

다음 방법을 통해 versions 매개변수와 prefix 요청 매개변수를 사용하여 특정 객체의 모든 버전을 조회할 수 있습니다. prefix에 대한 자세한 정보는 GET Bucket Object versions 문서를 참조하십시오.
특정 객체의 모든 버전 조회 요청 예시:

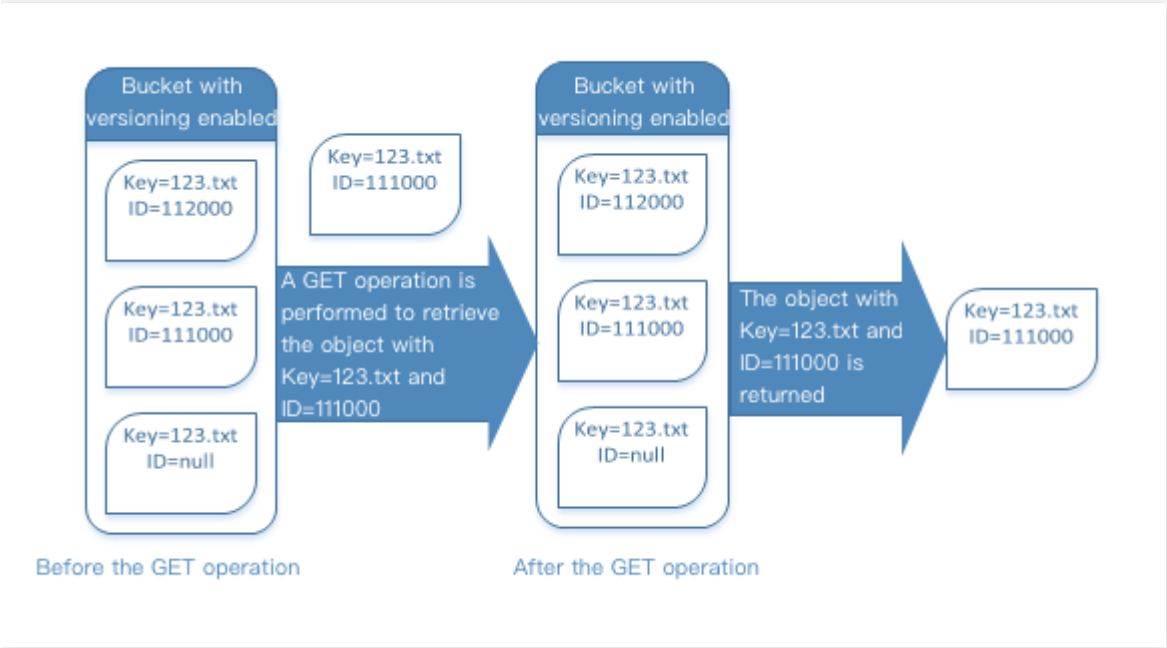
```
GET /?versions&prefix=ObjectKey HTTP/1.1
```

데이터 기본 버전 조회

GET 요청을 사용할 때 버전 ID를 지정하지 않는 경우 객체의 현재 버전을 조회합니다. 즉, 다음 이미지와 같이 GET 요청 시 123.txt 객체의 현재 버전(최신 버전)을 반환합니다.



GET 요청을 사용할 때 버전 ID를 지정하는 경우 지정 버전 ID의 객체를 조회합니다. 즉, 다음 이미지와 같이 GET versionId 요청 시 지정 버전(현재 버전 가능)의 객체를 조회합니다.



객체 버전의 메타데이터 조회

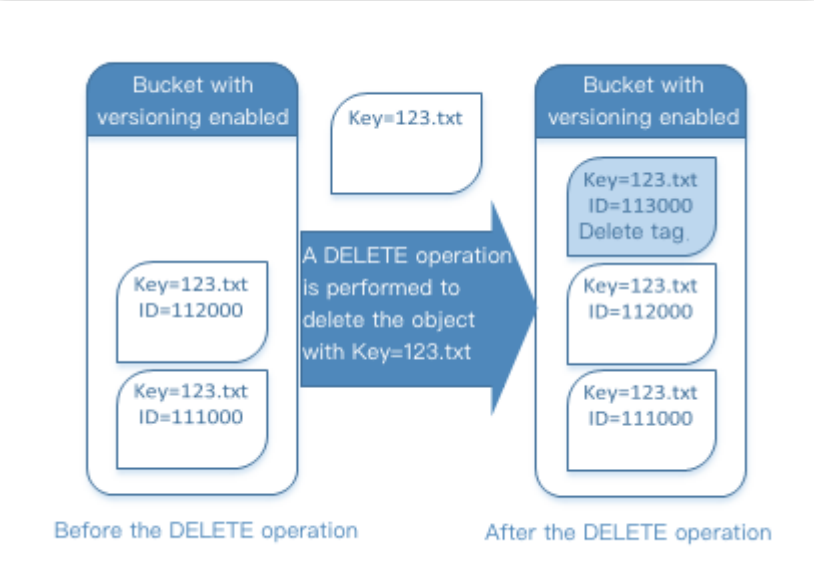
객체의 메타데이터(해당 콘텐츠가 아닌)를 조회할 경우, HEAD 작업을 사용하여 조회할 수 있습니다. 기본적인 상황에서 최신 버전의 메타데이터가 조회되며, 지정 객체 버전의 메타데이터를 조회하고 싶은 경우 요청 전송 시 해당 버전 ID를 지정해야 합니다.
지정 버전 객체의 메타데이터 조회 방법:

- versionId를 객체의 메타데이터를 조회할 버전 ID로 설정합니다.
- 지정한 versionId의 HEAD 작업 요청을 전송합니다.

객체 삭제

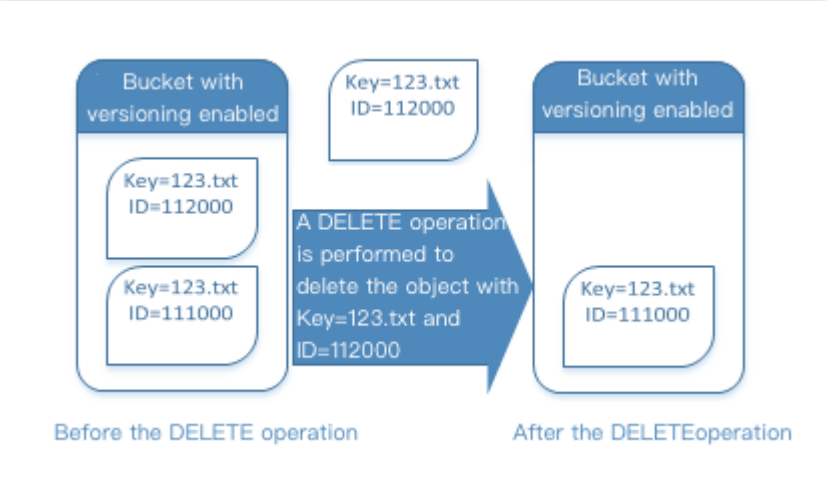
필요에 따라 언제든 불필요한 객체 버전을 삭제할 수 있습니다. 버전 제어를 활성화한 상태에서 DELETE 요청을 사용하는 경우 다음 두 가지 시나리오가 있을 수 있습니다.

1. 버전 ID를 지정하지 않고 일반적인 DELETE 작업을 실행하는 경우
   해당 작업 시나리오는 삭제 객체를 "휴지통"에 넣는 작업과 유사하지만 객체를 완전히 이동하지 않으며, 이후 사용자가 필요할 경우 데이터를 복구할 수 있습니다.
   다음 이미지와 같이 사용자가 DELETE 작업 시 버전 ID를 지정하지 않으면 실제로 Key=123.txt 객체를 삭제하지 않고 새로운 삭제 마커를 삽입하며 새로운 버전 ID를 추가합니다.



> 주의：
> COS는 버킷에서 삭제되는 객체에 새로운 버전 ID를 가진 삭제 마커를 삽입하며, 해당 삭제 마커는 삭제되는 객체의 현재 버전이 됩니다. 테스트로 해당 삭제 마커의 객체에 대해 GET 작업을 실행하면 COS는 해당 객체가 존재하지 않는다 인식하여 404 오류를 반환합니다.

2. 버전 ID를 지정하여 객체 버전을 삭제하는 경우(해당 시나리오는 버전 제어 객체를 영구적으로 삭제)



이전 버전으로 복원

버전 제어는 객체를 이전 버전으로 복원할 때 사용할 수 있으며, 다음과 같이 두 가지 방법이 있습니다.

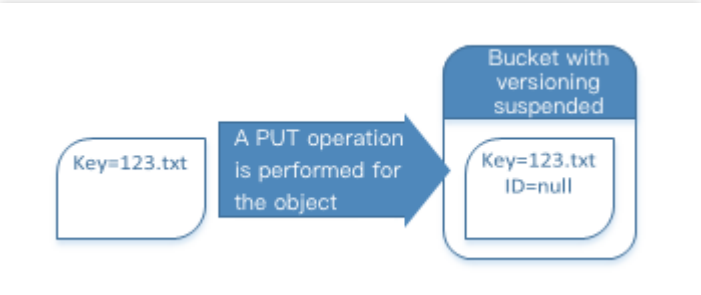
1. 객체 이전 버전을 동일한 버킷에 복사
   복사한 객체가 해당 객체의 현재 버전이 되며 모든 객체 버전이 보관됩니다.
2. 객체의 현재 버전 영구 삭제
   객체의 현재 버전을 삭제하면 이전 버전이 해당 객체의 현재 버전이 됩니다.

버전 제어 일시 중지 상태에서의 객체 관리

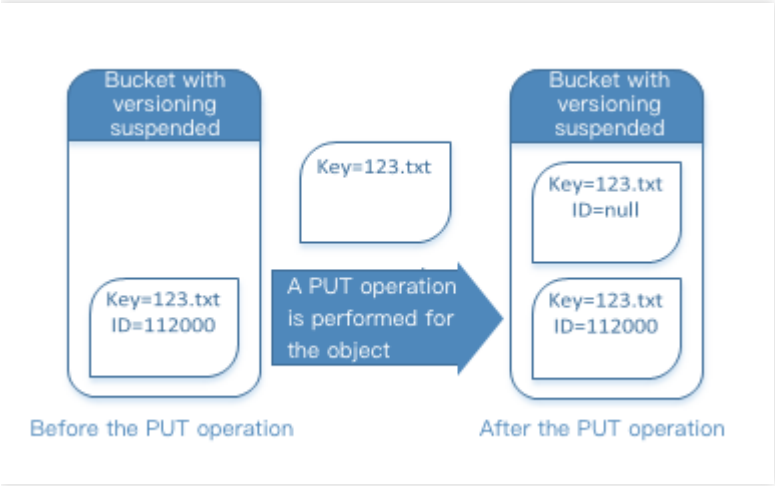
버전 제어 일시 중지 시 현재 버킷에 있는 객체는 변경되지 않으며, COS의 이후 요청에 대한 객체를 처리하는 방식이 변경됩니다. 버전 제어를 일시 중지한 버킷에서는 다음과 같이 객체를 관리합니다.

객체 업로드

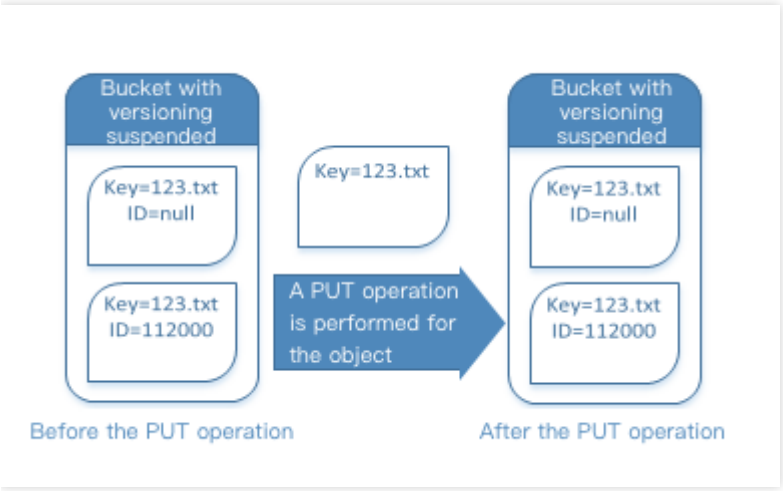
버킷에 버전 제어를 일시 중지한 후 사용자가 PUT, POST 또는 COPY 작업을 실행하면 COS는 다음 이미지와 같이 자동으로 버전 ID를 null로 하여 해당 버킷에 객체를 추가 저장합니다.



버킷에 버전 제어 객체가 존재하는 경우 다음 이미지와 같이 버킷에 업로드하는 객체가 현재 버전이 되며 버전 ID는 null이 됩니다.



버킷에 이미 null 버전이 존재하는 경우 다음 이미지와 같이 해당 null 버전을 덮어쓰며, 기존의 객체 콘텐츠가 상응하게 대체됩니다.



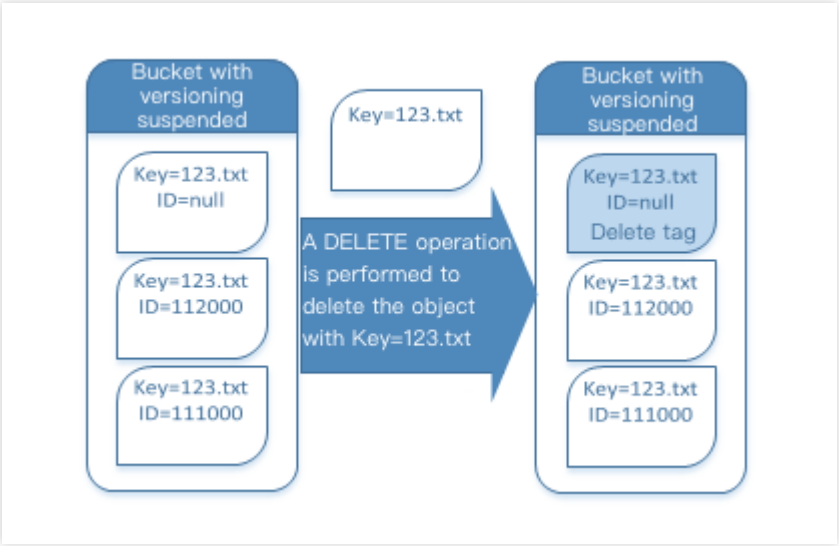
데이터 기본 버전 조회

버전 제어를 일시 중지한 버킷에 GET Object 요청을 전송하면 객체의 현재 버전이 반환됩니다.

객체 삭제

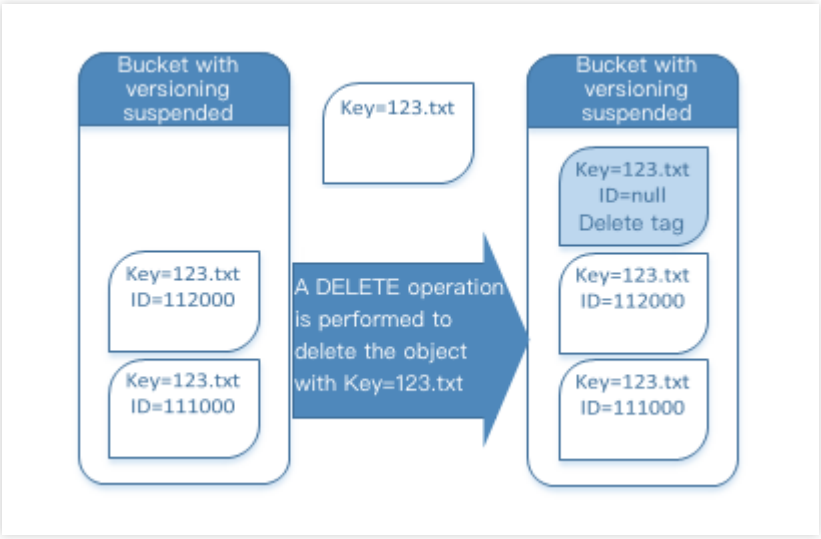
버전 제어를 일시 중지하고 DELETE 요청을 실행하는 경우 다음과 같은 상황이 있을 수 있습니다.

- 버킷에 null 버전의 객체가 존재하는 경우 버전 ID가 null인 객체를 삭제합니다.
  다음 이미지와 같이 일반적인 DELETE 작업을 실행하면 COS는 null 버전의 객체에 삭제 마커를 삽입합니다.

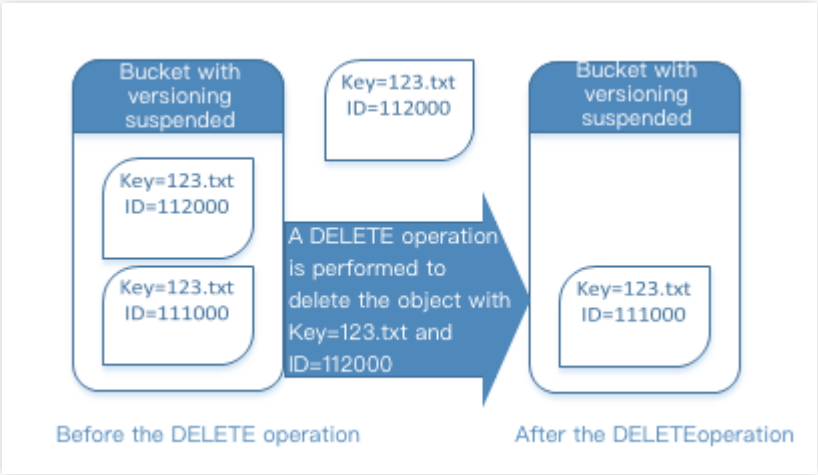


> 주의 :
> 삭제 마커는 콘텐츠가 존재하지 않으며, 삭제 마커가 null 버전으로 대체되는 경우 null 버전의 기존 콘텐츠가 유실됩니다.

- 버킷에 null 버전의 객체가 없는 경우 버킷에 삭제 마커가 추가됩니다.
  다음 이미지와 같이 버킷에 null 버전의 객체가 없을 때 DELETE 작업을 실행하면 COS는 어떠한 콘텐츠도 삭제하지 않고 삭제 마커만 삽입합니다.



- 버전 제어를 일시 중지한 버킷에 대해 루트 계정에서도 지정 버전을 영구 삭제할 수 있습니다.
  다음 이미지와 같이 객체 버전을 지정 삭제하여 영구적으로 해당 객체를 삭제할 수 있습니다.



주의 :

루트 계정 또는 루트 계정에서 권한을 부여한 계정만 객체 버전을 지정하여 삭제할 수 있습니다.

삭제 마커

최종 업데이트 날짜: : 2023-03-09 11:00:08

소개

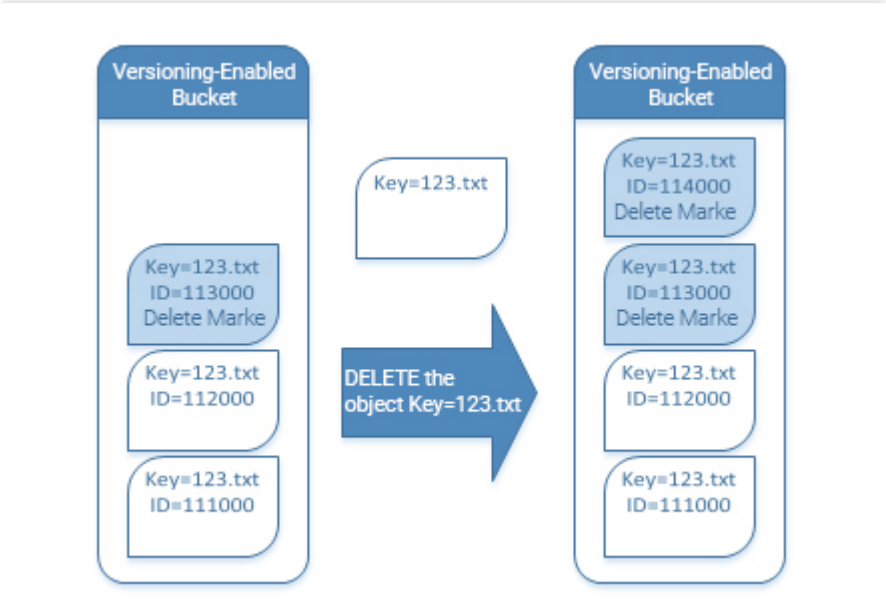
삭제 마커는 버전 제어 객체에 사용되며, COS에서 "객체가 이미 삭제"된 것으로 인식하게 만드는 표식입니다. 삭제 마커는 객체와 마찬가지로 객체 키(Key)와 버전 ID를 가지며, 차이점은 다음과 같습니다.

- 삭제 마커는 내용이 비어 있습니다.
- 삭제 마커에는 ACL 값이 존재하지 않습니다.
- 삭제 마커는 GET 요청 실행 시 404 오류를 반환합니다.
- 삭제 마커는 DELETE 작업만 지원합니다(루트 계정에서 요청해야 함).

"삭제 마커" 삭제

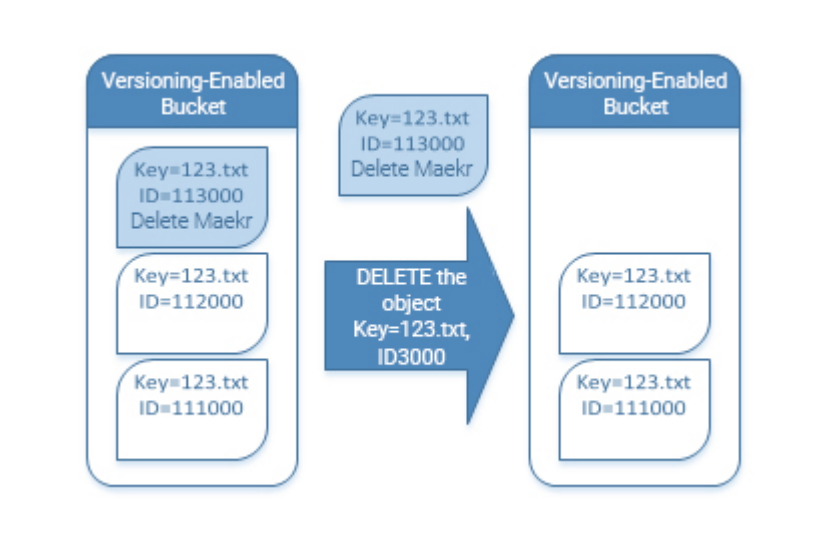
"삭제 마커"를 삭제할 경우 DELETE Object versionId 요청에서 해당 버전 ID를 지정하여 "삭제 마커"를 영구적으로 삭제할 수 있습니다. 삭제 마커의 버전 ID를 지정하지 않고 삭제 마커에 대한 DELETE 요청을 전달하는 경우 COS는 해당 삭제 마커를 삭제하지 않고 다시 새로운 삭제 마커를 추가합니다.

다음 이미지와 같이, 삭제 마커에 대해 일반적인 DELETE 요청을 실행하는 경우 어떠한 콘텐츠도 삭제되지 않고 버킷에 새로운 삭제 마커가 추가됩니다.



버전 제어를 활성화한 버킷에 추가된 삭제 마커는 고유 버전 ID를 가집니다. 따라서 버킷의 동일한 객체에는 여러 개의 삭제 마커가 존재할 수 있습니다. "삭제 마커"를 영구적으로 삭제하고 싶을 경우 반드시 DELETE Object versionId 요청에 해당 버전 ID가 포함되어 있어야 합니다.

다음 이미지와 같이 DELETE Object versionId 요청을 실행하여 "삭제 마커"를 영구적으로 삭제합니다.



> 루트 계정에서 `DeleteObject` 작업 권한을 부여해야만 "삭제 마커"를 삭제할 수 있습니다.

"삭제 마커" 영구 삭제 방법

1. versionId를 삭제 마커 버전 ID로 설정합니다.
2. DELETE Object versionId 요청을 발송합니다.

버킷 복사
버킷 복제 개요

최종 업데이트 날짜: : 2021-11-01 14:46:33

소개

크로스 버킷 복제를 통해 Cloud Object Storage(COS)는 한 버킷에서 다른 버킷으로 증분 객체를 자동으로 비동기식으로 복제할 수 있습니다. 크로스 버킷 복제를 통해 COS는 원본 버킷에서 타깃 버킷으로 객체 메타데이터 및 버전 ID와 함께 정확히 동일한 객체를 정확하게 복제할 수 있습니다. 또한 객체 추가 또는 삭제와 같은 객체 작업을 타깃 버킷과 동기화할 수도 있습니다.

주의 :

- 크로스 버킷 복제 기능을 활성화하려면 원본 버킷과 타깃 버킷에 버전 관리 기능이 활성화되어 있어야 합니다.
- 크로스 버킷 복제 기능 활성화 후, 데이터 복제 시 스토리지 유형을 설정하지 않는 한 모두 원본과 동일한 스토리지 유형으로 유지됩니다.
- COS 복제 시 원본 버킷의 액세스 제어 리스트(ACL)를 복제합니다. 원본 버킷과 타깃 버킷은 동일한 계정에서 소유해야 합니다.

적용 시나리오

- **원격 재해 복구:* *COS는 객체 데이터에 대해 '트웰브 나인' 수준의 가용성을 보장하지만, 여전히 전쟁 및 자연 재해와 같은 불가항력으로 인한 데이터 손실 가능성은 존재합니다. 다른 버킷에 별도의 복제본을 명시적으로 저장하여 데이터 손실을 방지하려면, 원격 재해 복구에 도움이 되는 크로스 버킷 복제를 사용할 수 있습니다. 특정 IDC에서 불가항력적인 요소로 인해 데이터가 유실된 경우 다른 버킷의 IDC에서 복제본 데이터를 제공합니다.
- **컴플라이언스 요건:* *COS는 기본적으로 물리적 디스크의 데이터에 대해 여러 복제본 및 삭제 코드를 제공하여 데이터 가용성을 보장합니다. 그러나 일부 산업에서는 복제본을 다른 버킷에 보관하도록 규정하는 규정 준수 요구사항이 있을 수 있습니다. 크로스 버킷 복제를 사용하면 버킷 간에 데이터를 복제하여 이러한 요구 사항을 충족할 수 있습니다.
- **액세스 딜레이 최소화:* *크로스 버킷 복제를 통해 최종 사용자가 다른 지역의 객체에 액세스하는 경우 가장 가까운 버킷에 객체 복제본을 유지할 수 있습니다. 이는 액세스 대기 시간을 최소화하여 더 나은 사용자 경험을 제공합니다.

- *특별한 기술 요구 사항:* 두 개의 서로 다른 버킷에 컴퓨팅 클러스터가 있고 클러스터가 크로스 버킷 복제를 사용하여 동일한 데이터 세트를 처리해야 하는 경우, 두 버킷에서 객체 복제본을 유지 관리할 수 있습니다.
- *데이터 마이그레이션 및 백업:* 비즈니스 확장 수요에 따라 비즈니스 데이터를 한 버킷에서 다른 버킷으로 복제하여 데이터 마이그레이션 및 데이터 백업을 실현할 수 있습니다.

주의 사항

복제 소요 시간

COS의 객체 복제 소요 시간은 객체의 크기, 버킷 리전 간의 거리, 객체의 업로드 방식 등의 요소에 의해 결정됩니다. 동기화 소요 시간은 해당 요소에 따라 달라지며, 몇 분에서 몇 시간까지 소요될 수 있습니다.

- 객체 크기: 대용량 객체 복제 시에는 더 많은 시간이 소요됩니다. 대용량 객체는 멀티파트 업로드 방식을 사용하여 객체의 업로드 및 동기화 시간을 단축할 것을 권장합니다.
- 버킷 리전 간의 거리: 리전 간의 거리가 멀수록 동기화 시 더 많은 데이터 전송 시간이 소요됩니다.
- 객체 업로드 방식: 간편 업로드 방식은 동시 작업이 불가능하며, 데이터를 하나씩 순차적으로 업로드 또는 다운로드합니다. 멀티파트 업로드 방식은 동시 작업이 가능하여 대용량 파일 업로드 시 멀티파트 업로드 방식을 이용하면 업로드 및 버킷 복제 작업을 더욱 빠르게 진행할 수 있습니다. 객체 업로드 방식에 대한 자세한 소개는 간편 업로드 및 멀티파트 업로드 문서를 참고하십시오.

라이프사이클 관련

크로스 버킷 복제를 사용하려면 먼저 버전 관리를 활성화해야 해야 하며, 이 경우 버킷에 객체의 여러 과거 버전이 유지되어 더 많은 스토리지 사용량이 발생합니다. COS 크로스 버킷 복제에는 요청, 다운스트림 트래픽 및 스토리지 사용량에 대한 요금이 발생합니다. 이 중 스토리지 사용량은 타깃 버킷 리전 요금으로 청구됩니다. 비용을 절감하거나 데이터 보존 방법을 사용자 정의하려면, 필요에 따라 라이프사이클 관리를 사용하십시오.

- 타깃 버킷의 객체 복제본이 원본 데이터와 동일한 라이프사이클 규칙을 따르도록 설정하려면, 타깃 버킷에 원본 버킷과 동일한 라이프사이클 규칙을 추가하십시오.
- 타깃 버킷에 라이프사이클 규칙을 설정한 경우, 크로스 버킷 복제로 생성된 객체 복제본의 생성 시간은 원본 객체 생성 시간이며, 복제본이 타깃 버킷에 복제된 시간이 아닙니다.
- 원본 버킷에 라이프사이클 규칙을 설정하였으며, 복제 중인 객체를 라이프사이클 규칙에 따라 삭제해야 하는 경우, 객체는 계속 복제되고 객체 복제본은 타깃 버킷에 유지됩니다.

버전 관리

버킷 복제 설정 시에는 원본 버킷과 타깃 버킷 모두에 버전 관리 기능을 설정해야 합니다. 버전 관리 기능에 대한 자세한 내용은 버전 제어 개요를 참고하십시오. 버전 관리 활성화 후, 버전 관리 비활성화 시 버킷 복제 기능에 대한 영향에 주의해야 합니다.

- 버킷 복제 기능을 활성화한 버킷에 버전 관리 기능을 비활성화하는 경우, COS에서 오류를 반환하며 먼저 버킷 복제 규칙 삭제 후 버전 관리 기능을 비활성화하도록 안내합니다.
- 타깃 버킷에 대한 버전 관리를 비활성화하려고 하면, COS에서 이러한 설정이 크로스 버킷 복제에 영향을 미친다는 메시지를 표시합니다. 버전 관리 비활성화를 진행하면 이 버킷을 타깃 버킷으로 하는 크로스 버킷 복제 규칙이 무효화됩니다.

복사 작용 설명

최종 업데이트 날짜: : 2021-03-02 14:54:51

본 문서에서는 버킷에 버킷 복사 기능 활성화 후 Cloud Object Storage(COS)에서 복사하는 콘텐츠와 복사하지 않는 콘텐츠에 대해 소개합니다.

##복사하는 콘텐츠

COS는 원본 버킷에 버킷 복사 기능을 활성화한 경우 다음 콘텐츠를 복사합니다.

- 버킷 복사 규칙 추가 후, 사용자가 원본 버킷에 새로 업로드한 모든 객체
- 객체의 메타데이터 및 버전 ID 등 객체 속성 정보
- 동일한 이름의 객체 신규 추가(객체 신규 추가와 동일), 객체 삭제 등 객체 관련 작업 정보

설명 :

- 원본 버킷에서 특정 객체 버전(즉, 버전 ID)을 지정하여 삭제하는 경우 해당 작업은 복사하지 않습니다.
- 원본 버킷에서 라이프사이클 규칙과 같은 버킷 레벨을 추가한 경우, 해당 설정으로 인한 객체 작업 또한 타깃 버킷에 복사되지 않습니다.

버킷 복사에서의 삭제 작업

원본 버킷에서 객체를 삭제하는 경우 버킷 복사는 다음과 같이 실행됩니다.

- 객체 버전 ID를 지정하지 않고 DELETE를 요청하는 경우 COS는 원본 버킷에 삭제 마커를 추가합니다. 이 때, 마커 삭제 동기화를 선택한 경우 버킷 복사 시 해당 마커를 타깃 버킷으로 복사하며, 마커 삭제 비동기화를 선택한 경우 타깃 버킷에는 삭제 마커가 추가되지 않습니다. 해당 두 상황에서 타깃 버킷의 해당 파일은 삭제되지 않으며, 사용자는 버전 ID를 지정해 객체의 이전 버전에 액세스할 수 있습니다. 버전 제어 및 삭제 마커에 대한 자세한 정보는 버전 제어 개요 문서를 참조하십시오.
- 객체 버전 ID를 지정하여 DELETE를 요청하는 경우 COS는 원본 버킷에서 해당 객체 버전을 삭제하고, 타깃 버킷에서는 해당 작업을 복사하지 않습니다. 즉, COS는 타깃 버킷에서 해당 객체 버전을 삭제하지 않으며, 이는 악성 데이터 삭제를 방지하기 위해서입니다.

복사하지 않는 콘텐츠

COS는 원본 버킷에 버킷 복사 기능을 활성화한 경우 다음 콘텐츠를 복사하지 않습니다.

- 버킷 복사 기능 활성화 전 이미 존재하고 있던 객체 콘텐츠, 즉 인벤토리 데이터
- 암호화된 객체의 암호화 정보, 즉 암호화 객체는 복사 후 암호화 정보가 유실됨
- 원본 버킷에 신규 추가된 데이터가 다른 버킷에서 복사된 객체 데이터인 경우

- 버킷 레벨의 설정 업데이트 행위
- 라이프사이클 설정 실행 후 결과

설명：

- 객체 데이터는 버킷 간의 버킷 복사 시 전달 복사되지 않습니다. 즉, A 버킷을 원본 버킷으로 하고 B 버킷을 타깃 버킷으로 하는 규칙과 B 버킷을 원본 버킷으로 하고 C 버킷을 타깃 버킷으로 하는 규칙, 해당 2개의 버킷 복사 규칙을 설정하는 경우 A 버킷에 새로 추가된 객체 데이터는 B 버킷에만 복사되고 C 버킷에는 복사되지 않습니다.
- 원본 버킷의 라이프사이클 설정을 업데이트하는 경우, COS에서는 해당 라이프사이클 설정을 타깃 버킷에 동기화하지 않습니다.
- 원본 버킷에만 라이프사이클을 설정한 경우 COS는 만료된 객체에 대해 삭제 마커를 추가하고 타깃 버킷에 해당 마커를 복사하지 않습니다. 타깃 버킷의 만료 객체를 삭제하고 싶은 경우, 타깃 버킷에 원본 버킷과 동일한 라이프사이클 규칙을 독립적으로 설정해야 합니다.

버킷 복제 구성

최종 업데이트 날짜: : 2022-09-28 14:50:08

적용 시나리오

버킷 복사 규칙 설정을 통해 객체 데이터를 원본 버킷에서 다른 특정 타깃 버킷으로 복사할 수 있습니다. 버킷 복사 기능은 원격 재해 복구, 업계 컴플라이언스 요건 충족, 데이터 마이그레이션 및 백업, 고객의 액세스 딜레이 단축, 서로 다른 리전의 클러스터에서의 편리한 데이터 액세스 등의 시나리오에 적용할 수 있습니다.

특수 시나리오:

- 다중 지역 백업: 원본 버킷에 대한 복사 규칙을 여러 개 설정하고 다른 리전의 버킷에 객체를 복사하여 다중 리전 백업을 수행할 수 있도록 지원합니다.
- 양방향 복사: 원본 버킷과 대상 버킷에 각각 복사 규칙을 생성하여 두 버킷의 양방향 복사를 지원하고 버킷 데이터의 동기화를 실현합니다.

> 주의 :
> 버전 관리 기능 활성화 후 새로 업로드하는 객체에 여러 버전이 생성되고 스토리지 용량을 차지합니다. 따라서 해당 버전의 객체에 대해서도 스토리지 요금이 발생합니다.

사용 방법

COS 콘솔 사용

COS 콘솔에서 버킷 복사 규칙을 설정할 수 있으며, 자세한 방법은 버킷 복제 설정 콘솔 가이드 문서를 참조하십시오.

REST API 사용

직접 REST API를 사용하여 버킷에 대한 버킷 복사 규칙을 설정하고 관리할 수 있습니다. 자세한 내용은 다음의 API 문서를 참조하십시오.

- PUT Bucket replication
- GET Bucket replication
- DELETE Bucket replication

SDK 사용

직접 SDK의 버킷 복사 방법을 호출할 수 있으며, 자세한 내용은 다음 각 언어별 SDK 문서를 참조하십시오.

- Android SDK
- C SDK
- C++ SDK
- .NET SDK
- Go SDK
- iOS SDK
- Java SDK
- JavaScript SDK
- Node.js SDK
- PHP SDK
- Python SDK

다중 AZ(MAZ) 기능 개요

최종 업데이트 날짜: : 2023-08-01 17:26:49

다중 AZ(Available Zone)는 COS에서 제공하는 다중 AZ 스토리지 아키텍처를 말하며, 데이터에 대한 IDC 수준의 재해 복구 기능을 제공할 수 있습니다.

귀하의 데이터는 한 리전의 여러 IDC에 분산되어 있습니다. 자연 재해나 정전과 같은 극한 상황으로 인해 IDC가 실패하더라도 다중 AZ 스토리지 아키텍처는 여전히 안정적이고 신뢰할 수 있는 스토리지 서비스를 제공할 수 있습니다.

다중 AZ는 99.9999999999%(12개 9) 설계 데이터 안정성과 99.995% 설계 서비스 가용성을 제공합니다. COS에 데이터 객체를 업로드할 때 스토리지 클래스를 지정하기만 하면 다중 AZ 리전에 저장할 수 있습니다.

설명 :

- 현재 COS의 MAZ 구성은 베이징, 광저우, 상하이, 홍콩, 싱가포르 리전에서만 지원되며 향후 다른 퍼블릭 클라우드 리전에서도 사용할 수 있습니다.
- MAZ 구성을 사용하면 높은 스토리지 요금이 발생합니다. 자세한 내용은 가격 | Cloud Object Storage를 참고하십시오.

다중 AZ의 장점

다중 AZ 리전에 데이터를 저장하면 데이터가 여러 청크로 분할되고 해당 코딩 청크는 삭제 코드 알고리즘을 기반으로 계산됩니다. 원본 데이터 청크와 코딩 청크는 혼합되어 스토리지 및 리전 내 재해 복구를 위해 리전의 다른 IDC에 균등하게 배포됩니다. IDC를 사용할 수 없게 되더라도 다른 IDC에서 정상적으로 데이터를 읽거나 쓸 수 있으므로 데이터가 손실 없이 지속적으로 저장되어 비즈니스 데이터 연속성과 고가용성이 유지됩니다. COS 다중 AZ 기능에는 다음과 같은 강점이 있습니다.

- **리전 내 재해 복구**: IDC 간 재해 복구가 지원됩니다. 다중 AZ 스토리지 아키텍처에서 객체 데이터는 동일한 리전의 서로 다른 IDC에 있는 서로 다른 장치에 저장됩니다. IDC가 실패하면 다른 중복 IDC를 계속 사용할 수 있으므로 비즈니스에 영향을 미치지 않고 데이터가 손실되지 않습니다.
- **안정성 및 내구성**: 삭제 코드 기반의 중복 저장 메커니즘을 활용하여 최대 99.9999999999%의 설계 데이터 안정성을 제공합니다. 데이터는 청크로 저장되고 동시에 읽고 쓰여 최대 99.995%의 설계 서비스 가용성을 제공합니다.
- **사용 용이성**: 객체 스토리지 클래스를 지정하여 데이터의 스토리지 아키텍처를 지정할 수 있습니다. 버킷의 객체를 선택하고 다중 AZ 아키텍처에 저장하여 사용하기 더 쉽게 할 수도 있습니다.

다중 AZ 스토리지와 비 다중 AZ 스토리지의 사양 및 제한 사항 비교는 아래와 같습니다.

비교 항목	MAZ 스토리지	비 MAZ 스토리지
설계된 데이터 내구성	99.9999999999% (12개 9)	99.999999999%(11개 9)
설계된 서비스 가용성	99.995%	99.99%
지원되는 지역	스토리지 클래스 개요를 참고하십시오	
지원되는 스토리지 클래스	MAZ_STANDARD MAZ_STANDARD_IA MAZ_INTELLIGENT_TIERING	STANDARD STANDARD_IA ARCHIVE DEEP ARCHIVE INTELLIGENT TIERING

사용 방법

버킷에 대해 MAZ를 활성화하고 버킷에 업로드된 객체에 대해 객체 스토리지 클래스를 MAZ로 설정할 수 있습니다. 객체를 업로드할 때 객체 스토리지 클래스를 지정하기만 하면 MAZ 스토리지 아키텍처에 저장할 수 있습니다.
즉, 다중 AZ 아키텍처에 파일을 저장하려면 다음 두 단계만 수행하면 됩니다.

1. 버킷 생성의 지침에 따라 버킷을 생성하고 **생성 시** MAZ 구성을 활성화합니다.
2. 파일을 업로드하고 업로드 중에 스토리지 클래스를 지정합니다. 파일 업로드 방법에 대한 자세한 내용은 객체 업로드를 참고하십시오.

설명：

- 버킷에 대해 다중 AZ를 활성화하면 비활성화할 수 없으므로 주의해서 활성화하십시오. 다중 AZ는 기본적으로 기존 버킷에 대해 활성화되지 않으며 새 버킷에 대해서만 활성화할 수 있습니다.
- MAZ 구성이 활성화된 버킷의 경우 객체를 MAZ 스토리지 클래스(MAZ_STANDARD, MAZ_STANDARD_IA 또는 MAZ_INTELLIGENT TIERING)에 업로드할 수 있습니다. 객체를 MAZ_INTELLIGENT TIERING에 업로드하려면 버킷에 대해 인텔리전트 티어링 구성도 활성화해야 합니다.
- 기존 데이터를 MAZ 버킷에 저장하려면 MAZ를 활성화한 상태에서 버킷을 생성하고 COS Batch의 일괄 복제 기능을 사용하여 기존 버킷의 파일을 새 버킷에 일괄 복제할 수 있습니다. COS Batch 사용 방법에 대한 자세한 내용은 일괄 작업을 참고하십시오.

사용 제한

현재 COS를 사용하면 MAZ_STANDARD, MAZ_STANDARD_IA 또는 MAZ_INTELLIGENT TIERING 스토리지 클래스에 객체를 업로드할 수 있습니다. 따라서 아래와 같이 스토리지 클래스 변경과 관련된 기능에 대한 제한 사항도 있습니다.

- 스토리지 클래스 제한: 현재 객체는 MAZ 스토리지 클래스(MAZ_STANDARD, MAZ_STANDARD_IA 또는 MAZ_INTELLIGENT TIERING)에만 업로드할 수 있습니다. MAZ_INTELLIGENT TIERING에 객체를 업로드하려면 버킷에 대해 MAZ 구성과 인텔리전트 티어링 구성을 모두 활성화해야 합니다.
- 작업 제한: 현재 객체는 업로드, 다운로드 및 삭제만 가능합니다. 객체는 다중 AZ 버킷에 복제할 수 있지만 단일 AZ 버킷에는 복제할 수 없습니다.
- 라이프사이클 제한: 현재 객체는 만료 시에만 삭제할 수 있지만 MAZ 스토리지 클래스에서 OAZ 스토리지 클래스로 전환할 수 없습니다.
- 리전 간 복제 제한: MAZ 스토리지 클래스에서 OAZ 스토리지 클래스로 객체를 복제할 수 없습니다.

데이터 보안
서버 암호화 개요

최종 업데이트 날짜: : 2023-05-25 17:34:12

개요

COS(Cloud Object Storage)는 데이터가 디스크에 기록되기 전에 객체 레벨에서 데이터를 암호화하고 데이터에 액세스할 때 자동으로 데이터를 해독합니다. 암호화 및 복호화는 서버에서 완료됩니다. 서버 측 암호화는 정적 데이터를 효과적으로 보호할 수 있습니다.

주의 :

- 암호화 객체와 비암호화 객체를 액세스할 때 경험적 측면에서는 차이가 없으나 객체에 대한 액세스 권한 보유 여부를 전제로 합니다.
- 서버 암호화는 객체 데이터 암호화만 지원하며 객체 메타데이터 암호화는 지원하지 않습니다. 서버로 암호화한 객체의 경우에는 반드시 익명 사용자가 아닌 유효한 서명으로 액세스해야 합니다.

적용 시나리오

- **기밀 데이터 스토리지 시나리오:** 서버 암호화로 저장할 기밀 데이터를 암호화하여 사용자의 개인 정보를 보호할 수 있습니다. 암호화 데이터는 사용자가 액세스할 때 자동으로 해독됩니다.
- **기밀 데이터 전송 시나리오:** COS는 기밀 데이터를 전송할 때 HTTPS로 SSL 인증서를 배포하여 암호화하는 기능을 제공합니다. 전송 링크 계층에 계층 암호화를 구현함으로써 데이터가 전송 과정에서 도난되거나 조작될 위험을 차단합니다.

암호화 방식

COS는 SSE-COS, SSE-KMS, SSE-C와 같은 다양한 서버 암호화 방식을 지원합니다. 사용자에게 적합한 암호화 방식을 채택하여 COS에 저장하는 데이터를 암호화할 수 있습니다.

SSE-COS 암호화

SSE-COS 암호화는 COS가 키를 호스팅하여 관리하는 서버 암호화 방식입니다. Tencent Cloud COS에서 마스터 키를 호스팅하고 데이터를 관리하며, 사용자는 COS를 통해 데이터를 직접 관리하고 암호화할 수 있습니다. SSE-COS

는 여러 요소로 강력한 암호화를 구현함으로써 모든 객체를 고유한 보안키로 암호화합니다. 또한 256비트의 고급 암호화 표준(AES-256)을 채택하여 데이터를 암호화하며, 정기적으로 마스터 키를 교체해 보안키 자체를 암호화합니다.

> 주의 :
>
> - `POST` 를 실행하여 객체를 업로드할 경우, `x-cos-server-side-encryption` 헤더가 아닌 테이블 필드에 동일한 정보를 제공해야 합니다. 자세한 내용은 POST Object를 참고하십시오.
> - 사전 서명한 URL로 업로드한 객체는 SSE-COS 암호화를 적용할 수 없습니다. COS 콘솔이나 HTTP 요청 헤더를 통해 서버 암호화를 실행해야 합니다.

COS 콘솔 사용

객체 암호화 설정 콘솔 문서를 통해 콘솔에서 객체에 SSE-COS 암호화하는 방법을 확인할 수 있습니다.

REST API 사용

> 주의 :
>
> - 버킷의 객체를 나열할 때 객체의 암호화 여부와 관계없이 리스트는 모든 객체의 리스트를 반환합니다.
> - POST를 실행하여 객체를 업로드할 경우, 해당 요청의 헤더가 아닌 테이블 필드에 동일한 정보를 제공하십시오. 자세한 내용은 POST Object를 참고하십시오.

사용자가 다음의 인터페이스를 요청하면 `x-cos-server-side-encryption` 헤더를 제공하여 서버 암호화를 적용합니다. 자세한 내용은 공통 요청 헤더 - SSE-COS를 참고하십시오.

- PUT Object
- Initiate Multipart Upload
- PUT Object - Copy
- POST Object

SSE-KMS 암호화

SSE-KMS 암호화는 KMS가 키를 호스팅하여 관리하는 서버 암호화 방식입니다. KMS는 Tencent Cloud가 선보인 보안 관리형 서비스로서 3rd party가 인증하는 하드웨어 보안 모듈(HSM, Hardware Security Module)로 보안키를 생성하여 보호합니다. KMS로 보안키를 간편하게 생성하고 관리할 수 있어 다양한 애플리케이션과 작업에 필요한 보안키 관리가 편리할 뿐만 아니라 모니터링과 컴플라이언스 니즈를 충족할 수 있습니다.

처음 SSE-KMS 암호화 방식을 사용할 경우, KMS 서비스 활성화가 필요합니다. KMS 서비스를 활성화하면 마스터 키(CMK)가 기본으로 생성됩니다. KMS 콘솔에서도 보안키를 직접 생성하고, 보안키 정책과 사용 방법을 정의할 수 있

으며, KMS를 사용하면 키 구성 요소의 출처를 **KMS** 또는 **외부**로 직접 선택할 수 있습니다. 자세한 내용은 Creating a Key와 Importing External Key를 참고하십시오.

주의：

- SSE-KMS는 객체 데이터에 한해 암호화하며, 객체의 메타데이터는 암호화 대상이 아닙니다.
- 현재 SSE-KMS는 베이징, 상하이, 중국홍콩, 광저우 리전에 한해 지원합니다.
- SSE-KMS 방식으로 암호화할 경우, KMS에서 부과하는 별도 요금이 발생합니다. 자세한 내용은 KMS 과금 개요를 참고하십시오.
- SSE-KMS 방식으로 암호화한 객체는 익명 사용자가 아닌 유효한 서명으로 액세스해야 합니다.

COS 콘솔 사용

객체 암호화 설정 콘솔 문서를 통해 콘솔에서 객체에 SSE-KMS 암호화하는 방법을 확인할 수 있습니다.

REST API 사용

주의：

- 버킷의 객체를 나열할 때 객체의 암호화 여부와 관계없이 리스트는 모든 객체의 리스트를 반환합니다.
- POST를 실행하여 객체를 업로드할 경우, 해당 요청의 헤더가 아닌 테이블 필드에 동일한 정보를 제공하십시오. 자세한 내용은 POST Object를 참고하십시오.

사용자가 다음의 인터페이스를 요청하면 `x-cos-server-side-encryption` 헤더를 제공하여 서버 암호화를 적용합니다. 자세한 내용은 공통 요청 헤더 - SSE-KMS를 참고하십시오.

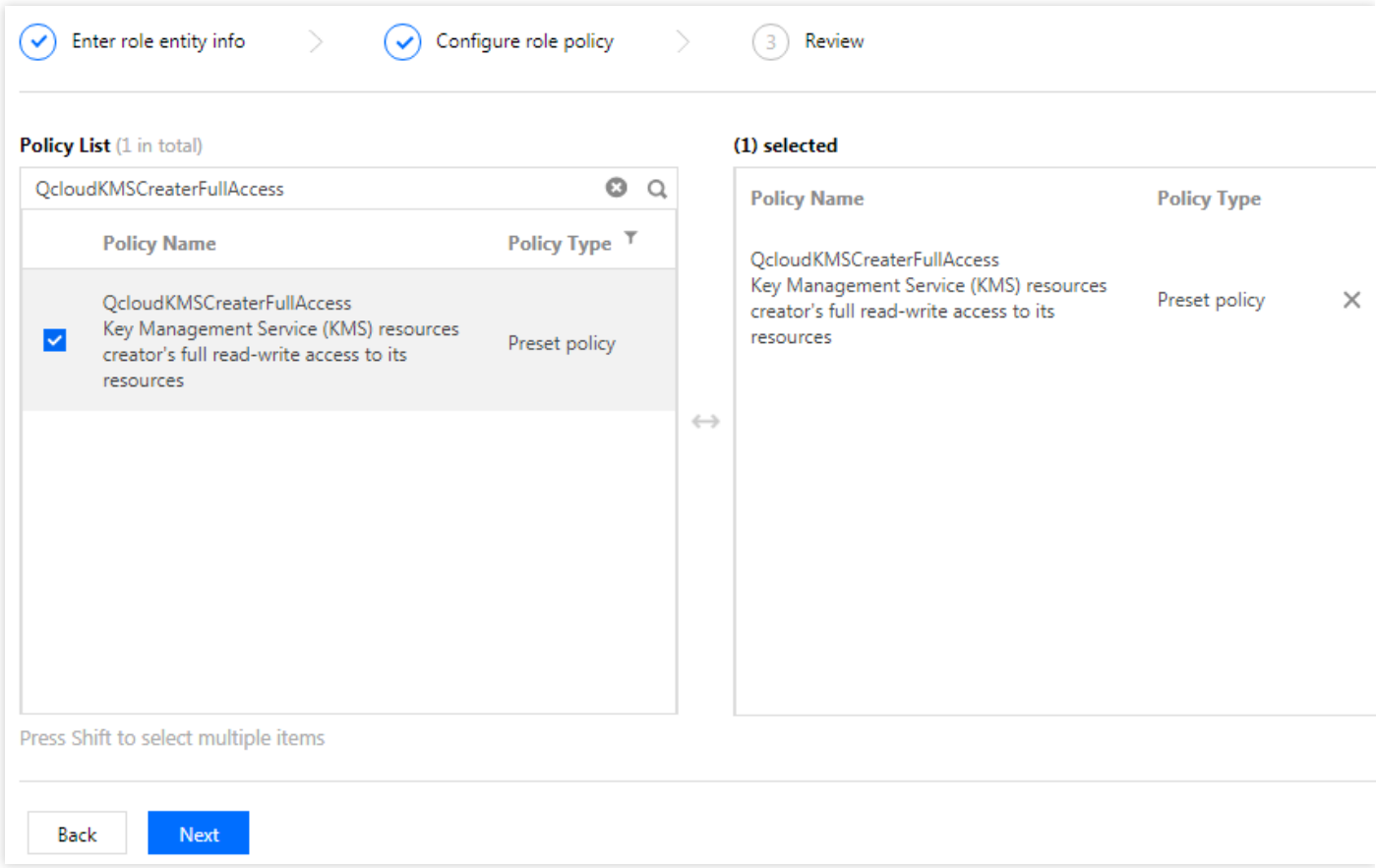
- PUT Object
- Initiate Multipart Upload
- PUT Object - Copy
- POST Object

주의사항

SSE-KMS 암호화에 **COS 콘솔**을 사용한 적이 없고 SSE-KMS 암호화에 **API**만 사용한 경우 먼저 CAM 역할을 생성해야 합니다. 자세한 생성 순서는 다음과 같습니다.

1. CAM 콘솔에 로그인하고 역할 페이지로 이동합니다.
2. **역할 생성**을 클릭하고 **Tencent Cloud 제품 서비스**를 역할 엔터티로 선택합니다.
3. 역할을 지원하는 서비스로 **COS**를 선택하고 **다음 단계**를 클릭합니다.

4. **QcloudKMSAccessForCOSRole** 역할 정책을 검색하여 선택한 후 **다음 단계**를 클릭합니다.



5. 역할 태그 키와 값을 설정하고 **다음 단계**를 클릭합니다.
6. 지정된 역할 이름(COS_QcsRole)을 입력합니다.
7. **완료**를 클릭하여 프로세스를 완료합니다.

SSE-C 암호화

SSE-C 암호화는 사용자 지정 키로 서버를 암호화하는 방식입니다. 암호화 키는 사용자가 직접 제공하는 것이며, 객체를 업로드할 때 COS는 사용자가 제공한 암호화 키로 사용자의 데이터에 AES-256 암호화를 적용합니다.

주의 :

- COS는 사용자가 제공한 암호화 키를 저장하지 않으며, 암호화 키를 저장할 때 임의 데이터의 HMAC 값을 추가합니다. HMAC 값은 사용자의 객체 액세스 요청을 검증하는 데 사용됩니다. COS에서 임의 데이터의 HMAC 값으로 암호화 키 값을 도출하거나 암호화된 객체의 콘텐츠를 복호화할 수 없습니다. 이 때문에 암호화 키를 분실하면 해당 객체를 다시 획득할 수 없습니다.
- POST를 실행하여 객체를 업로드할 경우, `x-cos-server-side-encryption-*` 헤더가 아닌 테이블 필드에 동일한 정보를 제공해야 합니다. 자세한 내용은 POST Object를 참고하십시오.
- SSE-C는 API를 통해서만 사용할 수 있으며, 콘솔에서는 지원하지 않습니다.

REST API 사용

사용자가 다음의 인터페이스를 요청할 때, PUT과 POST에 대한 요청은 `x-cos-server-side-encryption-*` 헤더 제공으로 서버를 암호화할 수 있습니다. GET과 HEAD에 대한 요청이 SSE-C로 암호화한 객체일 경우, `x-cos-server-side-encryption-*` 헤더를 제공하여 지정한 객체를 복호화해야 합니다. 자세한 내용은 공통 요청 헤더 -SSE-C를 참고하십시오. 다음은 지원하는 헤더 작업 항목입니다.

- GET Object
- HEAD Object
- PUT Object
- Initiate Multipart Upload
- Upload Part
- POST Object
- PUT Object - Copy

버킷 암호화 개요

최종 업데이트 날짜: : 2022-09-28 14:50:08

소개

버킷 암호화는 버킷의 설정 항목으로 버킷을 암호화하면 지정한 암호화 방식(기본)으로 버킷에 새로 업로드되는 모든 객체를 암호화할 수 있습니다.

현재 SSE-COS 암호화가 지원됩니다. 즉, COS(Cloud Object Storage)가 키를 호스팅하여 관리하는 서버 암호화 방식입니다.

서버 암호화와 관련한 자세한 내용은 서버 암호화 개요를 참조하십시오.

사용 방법

COS 콘솔 사용

COS 콘솔을 사용해 버킷 암호화를 설정할 수 있으며, 자세한 내용은 버킷 암호화 설정 콘솔 가이드 문서를 참조하십시오.

REST API 사용

다음 API로 버킷 암호화를 설정할 수 있습니다.

- PUT Bucket encryption
- GET Bucket encryption
- DELETE Bucket encryption

주의 사항

암호화 버킷에 객체 업로드

암호화 기능을 설정할 버킷에 관한 주의사항입니다.

- 버킷의 기존 객체에 대해서는 암호화가 적용되지 않습니다.
- 버킷 암호화 설정 후 버킷에 업로드하는 객체:
  - PUT 요청에 암호화 정보가 포함되지 않은 경우, 업로드 객체는 버킷 암호화 설정을 통해 암호화합니다.
  - PUT 요청에 암호화 정보가 포함된 경우, 업로드 객체는 PUT 요청에 있는 암호화 정보를 통해 암호화합니다.
- 버킷 암호화 설정 후 버킷에 전달하는 리스트 보고서:
  - 암호화되지 않은 리스트의 경우, 버킷 암호화 설정을 통해 전달하는 리스트를 암호화합니다.

  - 암호화된 리스트의 경우, 리스트 암호화 설정을 통해 전달하는 리스트를 암호화합니다.
- 버킷 암호화 설정 후, 버킷에 Origin-pull되는 데이터는 버킷 암호화 설정(기본) 방식으로 암호화합니다.

리전 간 복제 규칙을 설정한 버킷 암호화

리전 간 복제 규칙을 설정한 타깃 버킷에 버킷 암호화를 설정할 경우, 다음 사항을 주의하십시오.

- 소스 버킷의 객체가 암호화되지 않은 경우, 타깃 버킷의 복제 객체에 대해 기본 암호화 설정을 합니다.
- 소스 버킷의 객체가 암호화 상태이면 타깃 버킷의 복제 객체에 소스 버킷의 암호화를 상속하고, 별도의 버킷 암호화 설정 작업을 수행하지 않습니다.

객체 잠금 개요

최종 업데이트 날짜: : 2023-01-06 16:22:53

개요

Cloud Object Storage(COS)는 객체 잠금 기능을 제공합니다. 보존 중에 객체를 덮어쓰거나 삭제하지 못하도록 객체를 잠글 수 있습니다.

> 설명 :
>
> - 이 기능을 통해 COS는 전자 기록 보관에 대한 엄격한 요구 사항(SEC Rule 17a-4(f), FINRA 4511 및 CFTC 1.31 포함)을 충족할 수 있습니다.
> - SEC 규칙 17a-4는 1934년 미국 증권 거래법에 따라 미국 증권 거래 위원회에서 발행한 규정입니다. 이 규칙은 주식, 채권 및 선물과 같은 금융 유가 증권의 거래 또는 중개를 거래하는 회사의 데이터 보존, 인덱싱 및 접근성에 대한 요구 사항을 설명합니다. 이 규칙에 따르면 다양한 유형의 거래에 대한 기록을 보관해야 하며 즉시 액세스는 2년, 비즉시 액세스는 최소 6년 동안 재작성하거나 삭제할 수 없습니다.

객체 잠금은 버킷 레벨 기능입니다. 즉, 각 버킷에는 시간 기반 객체 잠금 규칙이 하나만 있을 수 있습니다. 이 기능을 활성화한 후에는 1일 - 100년 사이의 보존 기간이 필요합니다. 영구 보존은 허용되지 않습니다.

버킷에 객체 잠금을 설정하면 보관 주기 내에는 다음 작업을 할 수 없습니다.

- 객체를 삭제하거나 수정할 수 없습니다.
- 객체의 스토리지 클래스는 수정할 수 없습니다.
- HTTP 헤더 및 사용자 메타데이터(Content-Type, Content-Encoding, Content-Language, Content-Disposition, Cache-Control, Expires 및 x-cos-meta- 포함)는 수정할 수 없습니다.
- 객체 태그는 수정할 수 없습니다.

시간 기반 객체 잠금 규칙은 하나의 상태만 갖습니다. 즉, 규칙이 제출되면 적용됩니다. 규칙을 수정하거나 삭제할 수 없습니다. 보존 기간만 연장할 수 있습니다.

사용 방법

콘솔 또는 API를 사용하여 객체 잠금을 구성할 수 있습니다.

COS 콘솔 사용

COS 콘솔에서 객체를 잠그는 방법에 대한 정보는 버킷 복제 설정을 참고하십시오.

REST API 사용

다음 API를 직접 호출하여 객체 잠금을 관리할 수 있습니다.

- PUT Bucket ObjectLockConfiguration
- GET Bucket ObjectLockConfiguration
- GET Object Retention

제한 설명

1. 이제 객체 잠금은 얼로우 리스트에 있는 고객만 사용할 수 있습니다. 이 기능을 사용하려면 당사에 문의하기하십시오.

2. 활성화되면 객체 잠금 구성이 5초 이내에 버킷에 적용됩니다.

3. 객체 잠금은 아래와 같이 기존 버킷과 객체 모두에 대해 설정할 수 있습니다.
   2012년 7월 1일에 examplebucket이라는 버킷을 생성하고 서로 다른 시점에 세 개의 객체(test1.txt, test2.txt, test3.txt)를 업로드했다고 가정합니다(업로드 날짜는 다음 표에 표시됨). 그런 다음 2013년 9월 1일에 이 버킷에 대한 객체 잠금 규칙을 생성하여 객체를 5년 동안 보관했습니다. 이 경우 각 객체의 잠금 규칙 만료 날짜는 다음과 같습니다.

객체	업로드 날짜	객체 잠금 규칙의 만료 날짜
test1.txt	2012년 7월 1일	2017년 6월 30일
test2.txt	2013년 9월 1일	2018년 8월 31일
test3.txt	2017년 7월 30일	2022년 7월 29일

4. 객체 잠금이 활성화된 버킷에는 버전 관리가 지원되지 않습니다. 버킷에서 버전 관리 기능이 활성화되거나 일시 중단된 경우 객체 잠금도 활성화할 수 없습니다.

5. 객체 잠금이 활성화된 버킷에는 버킷 간 복제가 지원되지 않습니다. 버킷 간 복제 규칙에 따라 원본 및 대상 버킷에 버전 관리가 활성화되어 있어야 하며, 이는 객체 잠금이 활성화된 버킷에 대해 지원되지 않기 때문입니다.

6. INTELLIGENT TIERING은 객체 잠금이 활성화된 버킷에 대해 지원되지 않으며 객체 잠금은 INTELLIGENT TIERING이 활성화된 버킷에서 활성화할 수 없습니다.

7. 보존 기간 동안 잠긴 객체의 스토리지 클래스는 변환할 수 없습니다. 또한 객체 잠금은 라이프사이클 규칙과 호환되지 않습니다. 객체 잠금이 활성화된 버킷은 라이프사이클 규칙으로 구성할 수 없으며 그 반대의 경우도 마찬가지입니다.

8. 객체 잠금이 활성화된 경우 불완전한 다중 업로드는 객체 잠금 규칙의 적용을 받지 않으며 버킷에 대해 제거될 수 있습니다.

9. 객체 잠금은 일단 활성화되면 비활성화할 수 없습니다. 버킷의 모든 파일이 제거된 후 버킷을 삭제하여 객체 잠금 규칙을 취소할 수 있습니다.

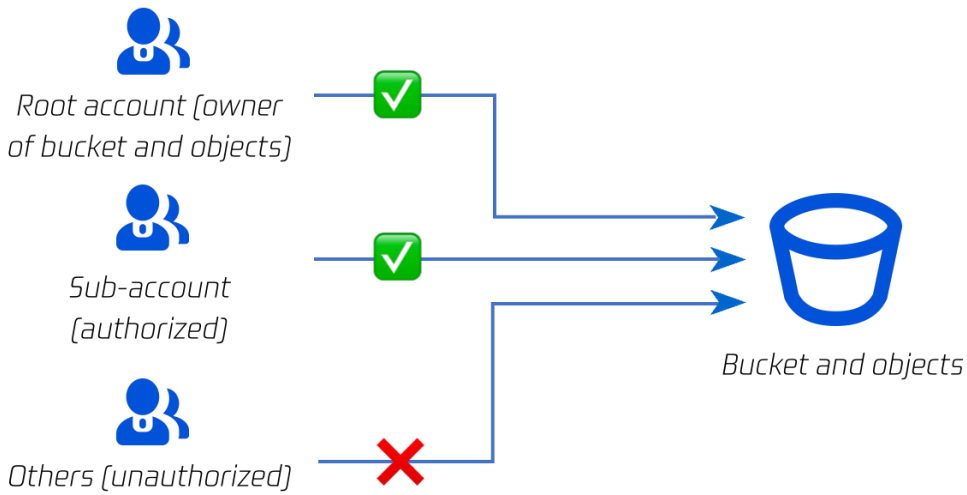
0. 객체 잠금이 활성화된 경우 버킷 및 객체의 ACL을 계속 수정할 수 있습니다.

액세스 관리
액세스 권한 설정 안내
액세스 제어 개요
액세스 제어 기본 개념

최종 업데이트 날짜: : 2022-11-14 14:28:36

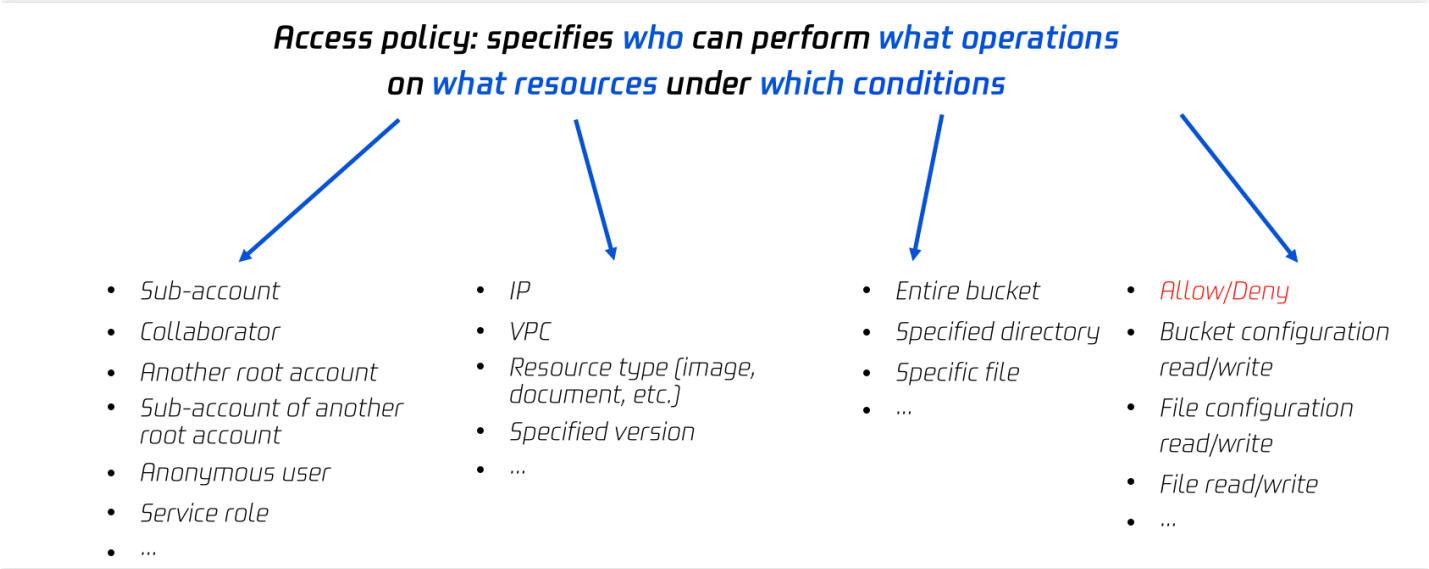
기본적으로 Tencent Cloud COS(Cloud Object Storage)의 리소스(버킷 및 객체)는 비공개입니다. Tencent Cloud 루트 계정(리소스 소유자)만 버킷과 객체에 액세스하고 수정할 수 있으며, 다른 사용자(서브 계정, 익명 사용자 등)는 권한 없이 URL을 통해 객체에 직접 액세스할 수 없습니다.

Tencent Cloud 서브 계정을 생성한 후 액세스 정책을 통해 서브 계정에 권한을 부여할 수 있으며, Tencent Cloud가 아닌 사용자에게 리소스를 공개해야 하는 경우 리소스(버킷, 객체, 디렉터리)에 대한 공개 권한(공개 읽기)을 설정하여 구현할 수 있습니다.



액세스 제어 요소

액세스 권한 부여는 누가, 어떤 조건 하에서, 어떤 리소스에 대해, 구체적인 작업을 실행할 제어 능력을 가질지에 대한 조합을 결정하는 것을 의미합니다. 따라서 액세스 권한에는 일반적으로 **자격, 리소스, 작업, 조건(선택 사항)** 총 네 가지 요소가 포함됩니다.



액세스 권한 요소

Tencent Cloud 자격(Principal)

사용자가 Tencent Cloud 계정을 신청하면 시스템에서는 Tencent Cloud 서비스에 로그인할 수 있는 루트 계정 자격을 생성합니다. Tencent Cloud 루트 계정은 사용자 관리 기능을 통해 다양한 유형의 사용자를 분류하여 관리합니다. 사용자 유형은 **협업 파트너, 정보 수신자, 서브 계정, 역할** 등으로 나뉘며, 자세한 정의는 CAM의 User Types 및 Glossary 문서를 참고하십시오.

> 설명：
> 사내 직원에게 권한을 부여하려면 CAM 콘솔에서 서브 계정을 생성한 후 버킷 정책, ACL 또는 사용자 정책 중 하나 이상을 사용하여 서브 계정에 대한 특정 권한을 설정해야 합니다.

COS 리소스(Resource)

Bucket과 Object는 COS의 기본 리소스이며, 그 중 폴더는 특별한 객체이며, 폴더를 통해 폴더 아래의 객체에 권한을 부여할 수 있습니다. 자세한 내용은 폴더 권한 설정을 참고하십시오.

또한 버킷과 객체 모두에 관련된 서브 리소스가 있습니다.

버킷의 서브 리소스는 다음을 포함합니다.

- acl과 policy: 버킷의 액세스 제어 정보
- website: 버킷의 정적 웹 사이트 호스팅 설정
- tagging: 버킷의 태그 정보

- cors: 버킷의 크로스 도메인 설정 정보
- lifecycle: 버킷의 라이프사이클 설정 정보

객체 서브 리소스는 다음을 포함합니다.

- acl: 객체의 액세스 제어 정보
- restore: 보관 유형 객체의 복구 설정

COS 작업(Action)

COS는 리소스에 대한 다양한 API 작업을 제공합니다. 자세한 내용은 Operation List 문서를 참고하십시오.

COS 조건(Condition, 옵션)

COS 조건은 vpc, vip와 같이 권한이 적용되는 조건을 나타냅니다. 자세한 내용은 Conditions를 참고하십시오.

개인 소유 원칙

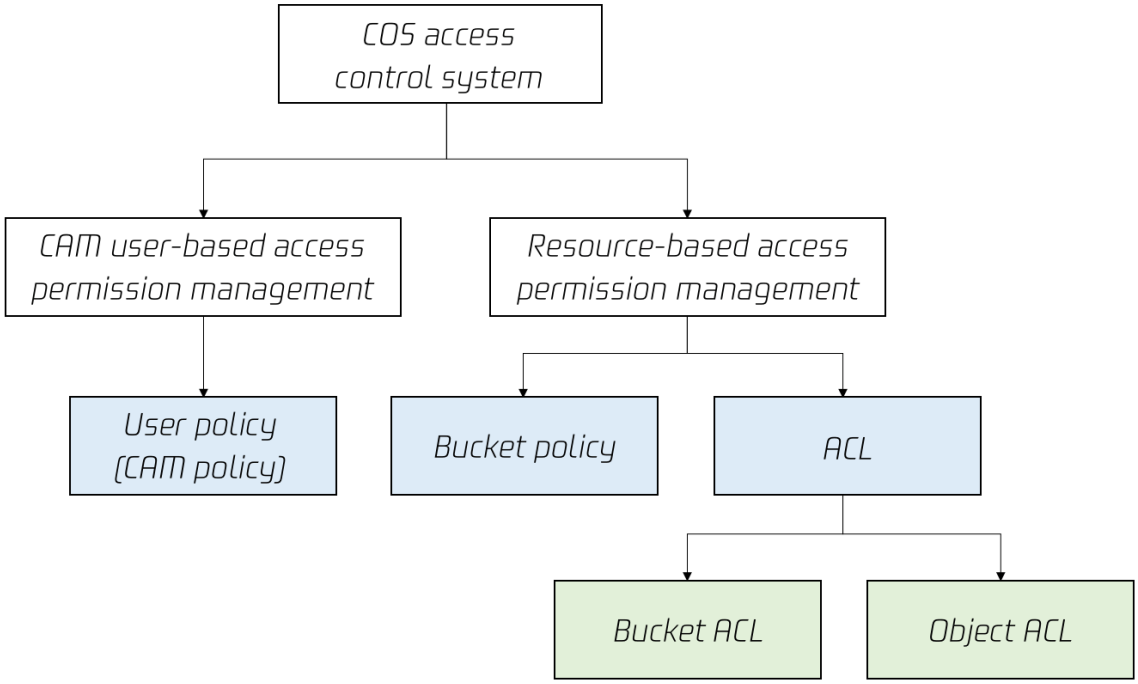
> 설명：
> Tencent Cloud COS 리소스는 기본적으로 비공개입니다.

- 리소스 소유자(버킷 리소스를 생성하는 Tencent Cloud 루트 계정)는 해당 리소스에 대한 최고 권한을 가지고 액세스 정책을 편집 및 수정할 수 있으며, 기타 사용자 또는 익명 사용자에게 액세스 권한을 부여할 수 있습니다.
- Tencent Cloud CAM(Cloud Access Management) 계정을 이용해 버킷을 생성하거나 객체를 업로드할 경우, 그 상위 루트 계정이 리소스 소유자가 됩니다.
- 버킷 소유자의 루트 계정은 다른 Tencent Cloud 루트 계정에 객체를 업로드할 수 있는 권한(교차 계정 업로드)을 부여할 수 있습니다. 이때 객체의 소유자는 여전히 버킷 소유자의 루트 계정입니다.

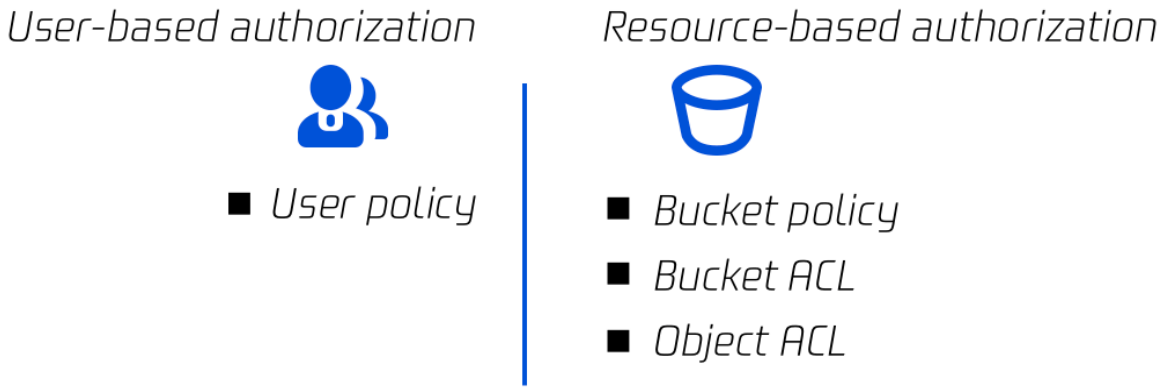
다양한 액세스 제어 경로

COS는 버킷 정책, 사용자 정책(CAM 정책), 버킷 ACL 및 객체 ACL을 포함하여 액세스 제어를 구현하기 위한 여러 권한 설정 방법을 제공합니다.

정책 설정의 시작점에 따라 리소스 기반과 사용자 기반의 두 가지 방법으로 나눌 수 있으며, 권한 부여 방법에 따라 정책과 ACL의 두 가지 방법으로 나눌 수 있습니다.

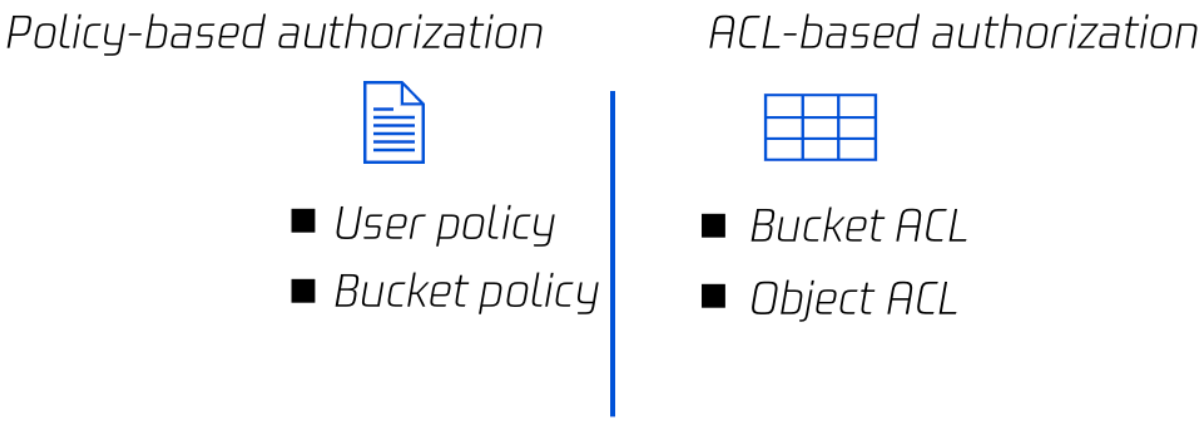


분류 방법1: 리소스 기반 vs 사용자 기반



- 리소스를 시작점으로: 버킷 정책, 버킷 ACL 및 객체 ACL을 포함한 특정 리소스와 권한을 연결하여 COS 콘솔에서 또는 COS API를 통해 설정합니다.
- 사용자를 시작점으로: 사용자와 권한을 연결하는 사용자 정책(CAM 정책)은 정책 작성 시 사용자를 입력할 필요가 없으며, 리소스, 작업, 조건 등을 지정하고 CAM 콘솔에서 설정해야 합니다.

분류 방법2: 정책 vs ACL



- 정책: 사용자 정책(CAM 정책) 및 버킷 정책은 완전한 정책 구문을 기반으로 권한이 부여되며, 권한 작업은 각 API에 해당하는 작업으로 세분화되며 허용/거부 효과 지정을 지원합니다.
- ACL: 버킷 ACL과 객체 ACL은 모두 액세스 제어 리스트(ACL)를 기반으로 구현됩니다. ACL은 정리되고 추상적인 권한에 해당하는 리소스와 관련된 지정된 피권한자 및 부여된 권한의 리스트입니다. 지정된 허용 효과만 지원합니다.

리소스 기반 정책

리소스 기반 정책에는 버킷 정책, 버킷 ACL 및 객체 ACL의 세가지가 포함됩니다. 액세스 제어는 **버킷** 및 **객체** 차원에서 지원됩니다. 자세한 내용은 다음 표를 참고하십시오.

규모	유형	설명 방식	지원 자격	지원 리소스 데이터 분할 정도	지원 작업	지원 효력
Bucket	액세스 정책 언어(Policy)	JSON	서브 계정, 역할, Tencent Cloud 서비스, 기타 루트 계정, 익명 사용자 등	버킷, 객체, 접두사 등	각각의 구체적인 작업	허용/명시적 거부
Bucket	액세스 제어 리스트(ACL)	XML	기타 루트 계정, 서브 계정, 익명 사용자	버킷	정리된 읽기/쓰기 권한	허용만 가능
Object	액세스 제어 리스트(ACL)	XML	기타 루트 계정, 서브 계정, 익명 사용자	객체	정리된 읽기/쓰기 권한	허용만 가능

버킷 정책(Bucket Policy)

버킷 정책(Bucket Policy)은 JSON 언어로 설명되며 익명 ID 또는 Tencent Cloud의 CAM 계정에 버킷 및 객체에 대한 액세스 및 작업 수행 권한 부여를 지원합니다. Tencent Cloud COS에서는 버킷 정책을 사용하여 버킷의 거의 모든 작업을 관리할 수 있습니다. ACL을 사용하여 설명할 수 없는 액세스 정책을 관리하려면 버킷 정책을 사용하는 것이 좋습니다. 자세한 내용은 버킷 정책을 참고하십시오.

> 주의 :
> Tencent Cloud 루트 계정은 그에 속한 리소스(버킷 포함)에 대해 가장 큰 권한을 가집니다. 버킷 정책이 거의 모든 작업을 제한하더라도, 루트 계정은 PUT Bucket Policy 작업 권한을 계속 가지고 있으므로, 해당 작업을 호출하여 버킷 정책을 점검하지 않도록 할 수 있습니다.

다음은 광저우 버킷 examplebucket-1250000000의 모든 객체에 대한 익명 사용자의 액세스를 허가하는 정책입니다. 서명 인증할 필요 없이 버킷의 모든 객체(GetObject)를 다운로드할 수 있고, URL을 아는 익명 사용자는 모두 객체를 다운로드할 수 있습니다(공개 읽기 방식과 유사).

```
{
"Statement":[
{
"Principal": "*",
"Effect": "Allow",
"Action": ["cos:GetObject"],
"Resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"]
}
],
"Version": "2.0"
}
```

액세스 제어 리스트(ACL)

ACL은 XML 언어로 설명됩니다. 리소스와 연결된 지정된 피부여자 및 부여된 권한 목록입니다. 각 버킷과 객체에는 익명 사용자 또는 다른 Tencent Clouds 루트 계정에 기본 읽기 및 쓰기 권한을 부여하는 연결된 ACL이 있습니다. 자세한 내용은 ACL을 참고하십시오.

> 주의 :
> 리소스 소유자는 전달한 ACL 중 항목 설명 여부에 상관없이 항상 리소스에 대한 FULL_CONTROL 권한을 가집니다.

다음은 버킷 ACL의 예시이며 버킷 소유자(사용자 UIN: 100000000001)의 전체 제어 권한을 설명합니다.

```
<AccessControlPolicy>
<Owner>
<ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
</Owner>
<AccessControlList>
<Grant>
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="RootAcco
unt">
<ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
</Grantee>
<Permission>FULL_CONTROL</Permission>
</Grant>
</AccessControlList>
</AccessControlPolicy>
```

다음은 객체 ACL의 예시입니다. 객체 소유자(사용자 UIN: 100000000001)의 완전한 권한 제어를 설명하며 모든 사용자에게 읽기 권한(익명 사용자는 공개 읽기)을 부여합니다.

```
<AccessControlPolicy>
<Owner>
<ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
</Owner>
<AccessControlList>
<Grant>
<Grantee>
<ID>qcs::cam::uin/100000000001:uin/100000000001</ID>
</Grantee>
<Permission>FULL_CONTROL</Permission>
</Grant>
<Grant>
<Grantee>
<URI>http://cam.qcloud.com/groups/global/AllUsers</URI>
</Grantee>
<Permission>READ</Permission>
</Grant>
</AccessControlList>
</AccessControlPolicy>
```

사용자 정책

사용자는 CAM에서 루트 계정에 속한 다양한 유형의 사용자에게 여러 가지 권한을 부여할 수 있습니다.

사용자 정책과 버킷 정책의 가장 큰 차이점은 사용자 정책은 효력(Effect), 작업(Action), 리소스(Resource), 조건(Conditon, 옵션)만 설명하며 자격(Principal)은 설명하지 않는다는 점입니다. 이 때문에 사용자 정책 작성이 완료되면

다시 서브 계정, 사용자 그룹 또는 역할 관련 작업을 실행해야 합니다. 사용자 정책은 익명 사용자에게 작업 또는 리소스 권한을 부여할 수 없습니다.

사전 설정된 정책을 연결하여 권한 부여하거나 사용자 정책 작성 후 지정된 ID와 연결하여 사용자 액세스를 관리할 수 있습니다. 자세한 내용은 사용자 정책을 참고하십시오.

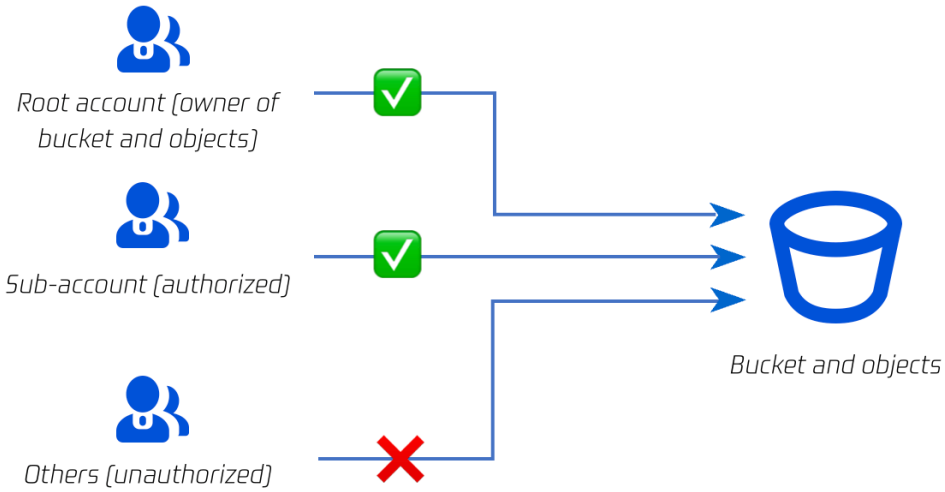
다음은 광저우에 위치한 버킷 examplebucket-1250000000의 모든 COS 작업에 권한 위임을 한 정책입니다. 정책을 저장한 후 CAM의 서브 계정, 사용자 그룹 또는 적용 가능한 역할과 연결해야 합니다.

```
{
"Statement":[
{
"Effect": "Allow",
"Action": ["cos:*"],
"Resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*",
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/"
]
}
],
"Version": "2.0"
}
```

COS 권한 부여 및 실명 인증 프로세스

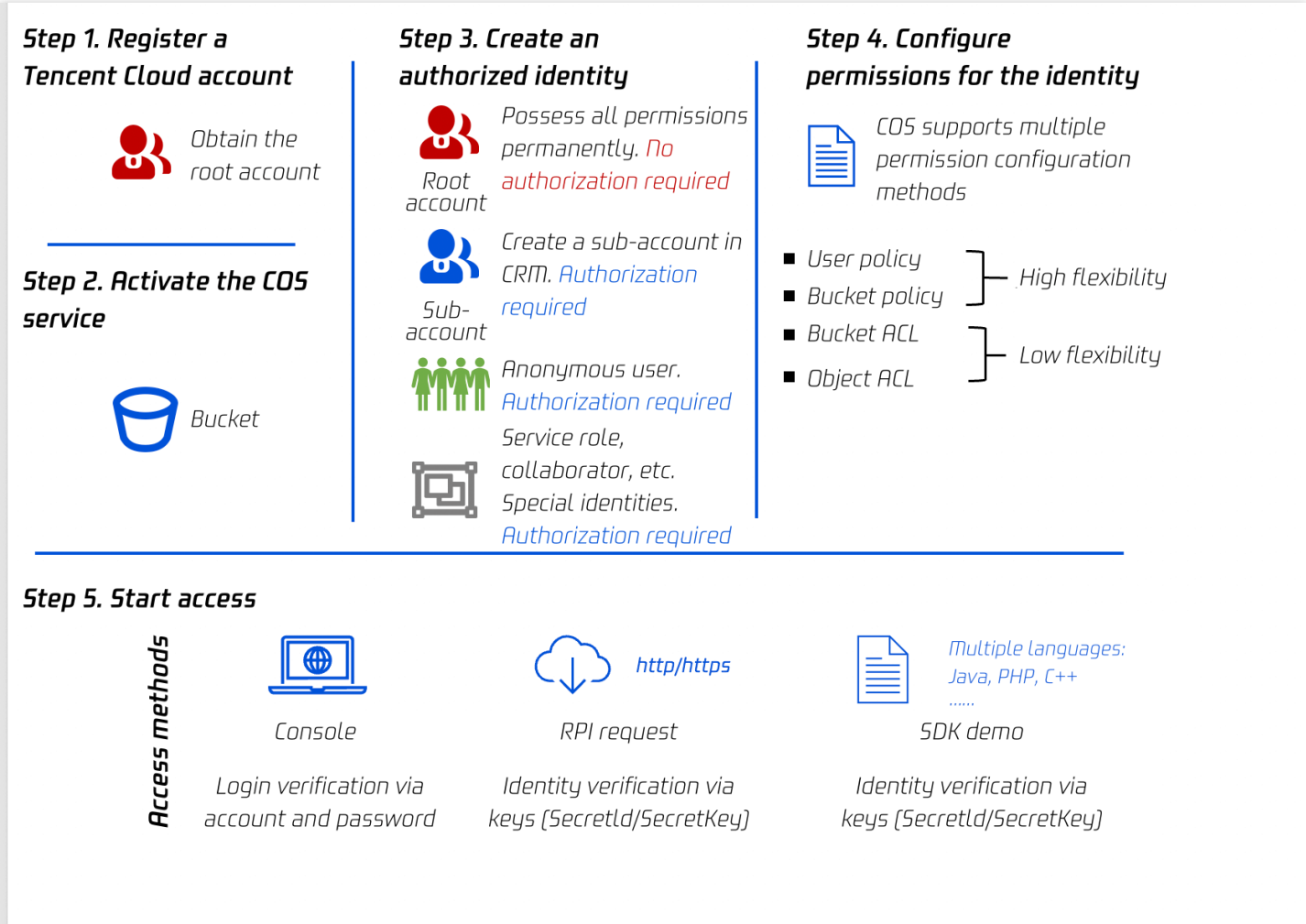
최종 업데이트 날짜: : 2022-05-16 16:49:33

기본적으로 COS(Cloud Object Storage)의 리소스(버킷 및 객체)는 개인 읽기/쓰기 이며, 객체 URL을 획득하더라도 익명의 사용자는 서명이 없기 때문에 url을 통해 리소스 콘텐츠에 액세스할 수 없습니다.



주요 단계

Tencent Cloud 계정 생성을 시작으로 COS의 권한 및 실명 인증 프로세스는 이 5단계를 거쳐야 합니다. Tencent Cloud 계정 생성, COS 서비스 활성화, 인증 ID 생성, ID에 대한 권한 설정 그리고 액세스 및 실명 인증을 시작합니다.



1단계: Tencent Cloud 계정 생성

Tencent Cloud 계정 생성 후 귀하의 계정은 루트 계정으로 존재하며 루트 계정이 가장 높은 권한을 가집니다.

2단계: COS 서비스 활성화

COS 서비스 활성화 후 생성하는 모든 버킷은 루트 계정이 소유합니다. 루트 계정은 모든 리소스를 사용할 수 있는 가장 높은 권한을 가지며, 서브 계정을 생성하고 서브 계정을 승인할 수 있는 권한이 있습니다.

3단계: 인증 ID 생성

> 주의 :
> 버킷이나 객체가 공개 읽기로 개방되어 있지 않는 한 COS에 액세스하려면 실명 인증 절차를 거쳐야 합니다.

루트 계정을 통해 여러 ID를 생성하고 리소스마다 다른 사용 권한을 부여할 수 있습니다.

- 회사 동료, 특정 부서의 사용자 등 특정 사용자에게 인증이 필요한 경우 CAM(Cloud Access Management) 콘솔에서 해당 사용자에 대한 서브 계정을 생성할 수 있습니다. 버킷 정책, 사용자 정책(CAM 정책), 버킷 ACL, 객체 ACL 등 다양한 인증 방법을 통해 서브 계정에 지정된 리소스에 대한 지정된 액세스 권한을 부여합니다.
- 다른 사람이 실명 인증 없이 url을 통해 직접 객체를 다운로드할 수 있도록 하는 등 익명 사용자에게 권한을 부여해야 하는 경우 리소스 권한을 기본 개인 읽기에서 공개 읽기로 수정해야 합니다.
- COS 버킷 사용 시 다른 Tencent Cloud 기타 서비스(예: CDN 등)가 필요한 경우에도 동일한 인증 절차를 따라야 하며, 귀하의 허가가 있는 경우 이러한 서비스는 서비스 역할을 통해 합법적으로 COS에 액세스할 수 있으며 CAM 콘솔에서 생성된 서비스 역할을 조회할 수 있습니다.

교차 Tencent Cloud 계정 인증의 경우 하나의 COS 버킷만 인증하려면, 버킷 정책 또는 버킷 ACL을 통해 다른 루트 계정을 직접 인증할 수 있습니다. 여러 COS 버킷 또는 여러 Tencent Cloud 리소스를 인증해야 하는 경우 CAM 콘솔을 통해 협업 파트너 ID를 생성하여 더 넓은 범위를 인증할 수 있습니다.

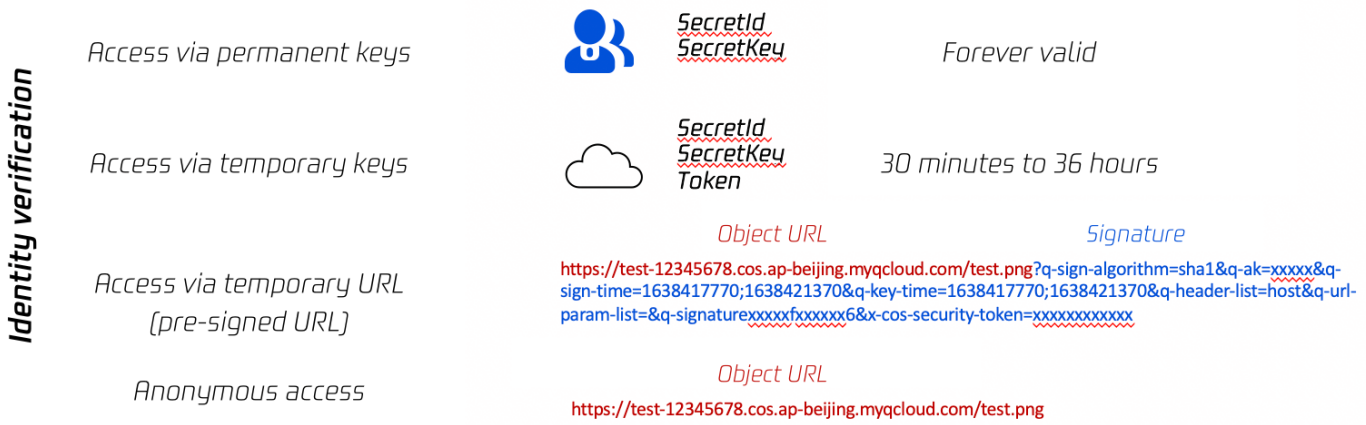
4단계: ID 권한 설정

COS는 버킷 정책, 사용자 정책(CAM 정책), 버킷 ACL 및 객체 ACL을 포함한 다양한 권한 설정 방식을 지원하며, 사용 시나리오에 따라 적절한 인증 방식을 선택할 수 있습니다.

5단계: 액세스 및 실명 인증 시작

콘솔, API 요청, SDK 등을 통해 COS에 액세스할 수 있습니다. 보안상의 이유로 버킷은 기본적으로 개인 읽기이며 어떤 방법을 사용하든 실명 인증이 필요합니다. 콘솔의 경우 계정 비밀번호를 사용하여 로그인합니다. API 요청과 SDK 모두에 대해 사용자는 키(SecretId/SecretKey)를 사용하여 실명 인증을 해야 합니다.

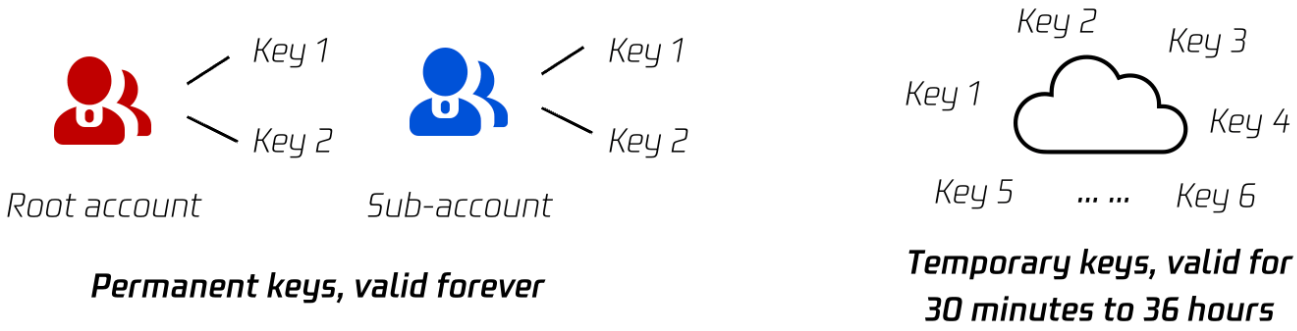
COS 실명 인증 방식



기본적으로 COS 버킷은 비공개이며 키(영구키, 임시키)를 통한 COS에 액세스 또는 사전 서명된 URL을 통한 액세스 모두 실명 인증을 완료해야 합니다. 특별한 경우에는 버킷을 공개 읽기로 개방할 수도 있으나 이는 위험한 작업입니다. 모든 사용자가 실명 인증 없이 객체 URL을 통해 직접 객체를 다운로드할 수 있습니다.

1. 영구 키를 사용하여 액세스

키(SecretId/SecretKey)는 사용자가 Tencent Cloud API에 액세스 시 인증할 때 사용하는 보안 자격 증명으로 API 키 관리에서 획득할 수 있습니다. 각 루트 계정과 서브 계정은 여러 개의 키를 생성할 수 있습니다.



영구 키는 SecretId와 SecretKey를 포함합니다. 각 루트 계정과 서브 계정은 두 쌍의 영구 키를 생성할 수 있습니다. 영구 키는 계정의 영구 ID를 나타내며 삭제하지 않으면 오랫동안 유효합니다. 자세한 내용은 영구 키를 사용하여 COS 액세스를 참고하십시오.

2. 임시 키를 사용하여 액세스

임시 키는 SecretId, SecretKey 및 Token을 포함하며, 각 루트 계정과 서브 계정은 여러 개의 임시 키를 생성할 수 있습니다. 임시 키는 영구 키에 비해 유효 기간이 있으며(기본 1800초) 최대 유효 기간은 루트 계정 7200초, 서브 계정 129600초로 설정할 수 있습니다. 자세한 내용은 연합 ID 임시 액세스 자격 증명 획득을 참고하십시오.

임시 키는 프런트 엔드 직접 전송과 같은 임시 인증 시나리오에 적합하며 영구 키에 비해 신뢰할 수 없는 사용자에게 임시 키를 배포하는 것이 더 안전합니다. 자세한 내용은 임시 키를 사용하여 COS 액세스를 참고하십시오.

3. 임시 URL(사전 서명된 URL)을 사용하여 액세스

자세한 내용 및 사용 설명은 사전 서명된 URL을 사용하여 COS 액세스를 참고하십시오.

객체 다운로드

타사가 버킷에서 객체를 다운로드할 수 있지만 상대방의 CAM 계정 또는 임시 키 사용을 제한하고 싶은 경우, URL 사전 서명 방식으로 타사에 서명을 제출함으로써 임시 다운로드 작업을 완료할 수 있습니다. 유효한 URL 사전 서명을 받은 계정은 모두 객체 다운로드를 진행할 수 있습니다.

- 콘솔 또는 COSBrowser에서 임시 다운로드 링크 가져오기(1 - 2시간 동안 유효)
  콘솔이나 COSBrowser에서 직접 객체의 임시 다운로드 링크를 빠르게 가져올 수 있으며, 브라우저에 직접 임시 링크를 입력하여 객체를 다운로드할 수 있습니다. 자세한 내용은 임시 링크 빠르게 획득 문서를 참고하십시오.
- SDK 사용하여 사전 서명 url 생성
  SDK를 사용하면 사용자 정의 유효 기간이 있는 사전 서명된 URL을 일괄적으로 가져올 수 있으며, 자세한 내용은 SDK 사용하여 사전 서명된 URL 일괄 가져오기 문서를 참고하십시오.
- 서명 툴을 사용하여 사전 서명된 URL 생성
  프로그래밍에 익숙하지 않은 사용자에게 적합한 사용자 정의 유효 기간의 사전 서명 URL 가져오기 입니다. 자세한 내용은 서명 툴 사용 문서를 참고하십시오.
- 서명 링크 자체 스티칭
  사전 서명된 URL은 실제로 객체 URL 뒤에 스티칭되는 서명입니다. 따라서 SDK, 서명 생성 툴 등을 통해 직접 서명을 생성하고 URL과 서명을 서명 링크로 스티칭할 수도 있습니다. 그러나 서명 생성 알고리즘의 복잡성으로 인해 이 사용 방식은 일반적으로 권장되지 않습니다.

객체 업로드

서드 파티가 버킷에 객체를 업로드할 수 있지만 CAM 계정 혹은 임시 키 사용을 원치 않는 경우 임시 업로드 작업을 완료하기 위해 URL 사전 서명을 사용해 서드 파티에게 서명을 제출합니다. 유효한 서명 URL을 받은 계정은 모두 객체 업로드를 진행할 수 있습니다.

- 방법1: SDK를 사용하여 사전 서명된 URL 생성
  각 언어 SDK는 사전 서명된 URL을 생성하고 업로드하는 방법을 제공하며, 생성 방법은 사전 서명된 인증 업로드를 참고하여 익숙한 개발 언어를 선택하십시오.
- 방법2: 서명 링크 자체 접합
  사전 서명된 URL은 실제로 객체 URL 뒤에 스티칭되는 서명입니다. 따라서 SDK, 서명 생성 툴 등을 통해 직접 서

명을 생성하고 URL과 서명을 서명 링크로 스티칭하여 객체 업로드를 할 수도 있습니다. 그러나 서명 생성 알고리즘의 복잡성으로 인해 이 사용 방식은 일반적으로 권장되지 않습니다.

4. 익명 액세스

기본적으로 COS 버킷은 비공개로 되어 있으며, 키(영구 키, 임시 키)를 통한 COS 액세스 또는 사전 서명된 URL을 통한 액세스 모두 실명 인증 절차를 거쳐야 합니다.

특별한 필요에 의해 버킷이나 객체를 공개 읽기로 개방할 수도 있으며 모든 사용자는 실명 인증 없이 객체 URL을 통해 객체를 직접 다운로드할 수 있습니다.

> 주의：
> 리소스를 공개 읽기로 개방하는 것은 보안상 위험하며, 리소스 링크가 유출되면 누구나 액세스할 수 있어 악의적인 사용자에게 트래픽을 도난당할 수 있습니다.

공개 읽기로 버킷 개방

콘솔에서 전체 버킷을 공개 읽기로 설정할 수 있습니다. 이 경우 버킷의 각 객체는 모두 객체 URL을 통해 직접 다운로드될 수 있습니다. 설정 방법은 버킷 액세스 권한 설정을 참고하십시오.

공개 읽기로 객체 개방

콘솔에서 개별 객체를 공개 읽기로 설정할 수 있습니다. 이 경우 URL을 통해 그 객체만 직접 다운로드될 수 있으며 다른 객체는 영향을 받지 않습니다. 설정 방법은 객체 액세스 권한 설정을 참고하십시오.

폴더를 공개 읽기로 개방

콘솔에서 폴더를 공개 읽기로 설정할 수 있습니다. 이 경우 폴더 아래의 모든 객체는 URL을 통해 직접 다운로드될 수 있으며 폴더 외부의 객체는 영향을 받지 않습니다. 설정 방법은 폴더 권한 설정을 참고하십시오.

최소 권한의 원칙 설명

최종 업데이트 날짜: : 2023-03-14 17:08:04

개요

Cloud Object Storage(COS)를 사용 시 임시 키를 사용해 사용자에게 해당 리소스의 작업 권한을 부여하거나 서브 계정 또는 협업 파트너에게 적합한 사용자 정책을 부여해, 그들이 COS의 리소스를 작업할 수 있도록 허용해야 할 수 있습니다. 또한 버킷에 관련 버킷 정책을 추가해 지정된 사용자가 버킷에서 지정된 작업을 하거나 지정된 리소스를 작업하도록 해야 할 수 있습니다. 이와 같은 권한 설정 시, 반드시 **최소 권한 원칙**을 준수해 데이터 자산 보안을 확보하십시오.

최소 권한 원칙이란, 권한을 부여할 때 권한 범위를 명확히 하여 **지정 사용자**에게 **어떤 조건**에서 **어떤 작업**을 수행하고 **어떤 리소스**에 액세스할지에 대해 권한을 명확하게 부여하는 것을 의미합니다.

주의 사항

사용자가 지정된 작업(예: `action:GetObject`)만 수행하거나 지정된 리소스(예: `resource:examplebucket-1250000000/exampleobject.txt`)에 액세스할 수 있도록 최소 권한 원칙을 엄격히 따르는 것이 좋습니다.
과도한 권한으로 인한 예기치 않은 무단 작업으로 인한 데이터 보안 위험을 방지하려면 사용자에게 모든 리소스(예: `resource:*`)에 대한 액세스 권한을 부여하거나 모든 작업(예: `action:*`)을 수행하지 않는 것이 좋습니다.

잠재된 데이터 보안 리스크 예시는 다음과 같습니다.

- 데이터 유출: 사용자에게 `examplebucket-1250000000/data/config.json` 및 `examplebucket-1250000000/video/` 와 같은 지정된 리소스를 다운로드할 수 있도록 권한을 부여하고 싶지만 권한 정책에 `examplebucket-1250000000/*` 을 포함하면 버킷의 모든 객체는 승인 없이 다운로드될 수 있어 예기치 않은 데이터 유출이 발생할 수 있습니다.
- 데이터 덮어쓰기: 사용자에게 `examplebucket-1250000000/data/config.json` 및 `examplebucket-1250000000/video/` 를 업로드할 수 있는 권한을 부여하고 싶지만 권한 정책에 `examplebucket-1250000000/*` 을 포함하면 승인 없이 버킷의 모든 객체를 업로드할 수 있으므로 의도하지 않은 객체를 덮어쓸 수 있습니다. 이러한 위험을 방지하기 위해 최소 권한 원칙을 따르는 것 외에도 버전 제어 개요에 설명된 대로 추적을 위해 모든 버전의 데이터를 유지할 수 있습니다.
- 권한 유출: 사용자가 버킷(`cos:GetBucket`)의 객체를 나열할 수 있는 권한을 부여하고 싶지만 권한 정책에서 `cos:*` 를 구성하면 버킷 재승인, 객체 삭제 및 버킷 삭제를 포함하여 버킷에 대한 모든 작업이 허용되므로 데이터가 매우 위험해집니다.

사용 가이드

최소 권한 원칙에 따라 정책에서 다음 정보를 명확히 지정해야 합니다.

- 위탁자(principal): 권한을 부여할 서브 계정(사용자 ID 필요), 협업 파트너(사용자 ID 필요), 익명 사용자 또는 사용자 그룹을 지정해야 합니다. 액세스에 임시 키를 사용하는 경우에는 필요하지 않습니다.
- 명령(statement): 해당 매개변수를 입력합니다.
  - 효력(effect): 정책이 allow인지 deny인지 지정해야 합니다.
  - 작업(action): 허용 또는 거부할 동작을 지정해야 합니다. 하나의 API 작업 또는 일련의 API 작업일 수 있습니다.
  - 리소스(resource): 권한이 부여된 리소스를 지정해야 합니다. 리소스는 6개 세그먼트 형식으로 설명됩니다. 리소스를 특정 파일(예: `exampleobject.jpg`) 또는 디렉터리(예: `examplePrefix/*`)로 설정할 수 있습니다. 필요한 경우가 아니면 `*` 와일드카드를 사용하여 모든 리소스에 대한 액세스 권한을 사용자에게 부여하지 마십시오.
  - 조건(condition): 정책의 효력이 발생하는 규제 조건을 기술합니다. 조건에는 오퍼레이터, 작업 키와 작업 값 구성이 포함됩니다. 조건 값은 시간, IP 주소 등의 정보를 포함합니다.

임시 키 최소 권한 가이드

임시 키 신청 과정에서 권한 정책 Policy 필드 설정을 통해 작업 및 리소스를 제한하고, 권한을 지정된 범위로 제한할 수 있습니다. 임시 키 생성 관련 자세한 설명은 임시 키 생성 및 사용 가이드 문서를 참고하십시오.

권한 부여 예시

Java용 SDK를 사용하여 지정된 객체에 액세스할 수 있는 사용자 권한 부여

Java SDK를 사용하여 사용자에게 `examplebucket-1250000000` 버킷의 `exampleObject.txt` 객체를 다운로드할 수 있는 권한을 부여하려면 구성 코드는 다음과 같아야 합니다.

```
// github에서 제공하는 maven 통합 방법에 따라 java sts sdk 가져오기
import java.util.*;
import org.json.JSONObject;
import com.tencent.cloud.CosStsClient;

public class Demo {
public static void main(String[] args) {
TreeMap<String, Object> config = new TreeMap<String, Object>();

try {
String secretId = System.getenv("secretId");//사용자 SecretId. 리스크를 줄이기 위해 서브 계정 키를 사용하고 최소 권한 원칙을 따르는 것이 좋습니다. 서브 계정 키를 가져오는 방법에 대한 자세한 내용은 다음을 참고하십시오. https://cloud.tencent.com/document/product/598/37140
String secretKey = System.getenv("secretKey");//사용자 SecretKey. 리스크를 줄이기 위
```

```
해 서브 계정 키를 사용하고 최소 권한 원칙을 따르는 것이 좋습니다. 서브 계정 키를 가져오는 방법에 대한 자세한 내용은 다음을 참고하십시오. https://cloud.tencent.com/document/product/598/37140
// 자신의 SecretId로 교체
config.put("SecretId", secretId);
// 자신의 SecretKey로 교체
config.put("SecretKey", secretKey);

// 임시 키의 유효 기간(초), 기본값: 1800; 최대값: 7200
config.put("durationSeconds", 1800);

// 자신의 bucket으로 교체
config.put("bucket", "examplebucket-1250000000");
// bucket 소재 리전으로 교체
config.put("region", "ap-guangzhou");

// 허용되는 경로 접두사(예: a.jpg, a/* 또는 *)로 변경합니다. 로그인 상태에 따라 업로드 경로를 결정할 수 있습니다.
// '*'를 입력하면 사용자가 모든 리소스에 액세스할 수 있습니다. 업무에 필요한 경우가 아니면 최소 권한 원칙에 따라 필요한 제한된 권한만 사용자에게 부여합니다.
config.put("allowPrefix", "exampleObject.txt");

// 키 권한 리스트. 단순 업로드, 양식을 이용한 업로드 및 멀티 파트 업로드에 필요한 권한은 다음과 같습니다. 기타 권한 리스트는 다음을 참고하십시오. https://cloud.tencent.com/document/product/436/31923
String[] allowActions = new String[] {
// 데이터 다운로드
"name/cos:GetObject"
};
config.put("allowActions", allowActions);

JSONObject credential = CosStsClient.getCredential(config);
// 성공하면 아래와 같이 임시 키 정보가 반환되어 출력됩니다
System.out.println(credential);
} catch (Exception e) {
//실패 시 예외가 발생합니다
throw new IllegalArgumentException("no valid secret !");
}
}
}
```

API를 사용하여 지정된 객체에 대한 액세스 권한 부여

API를 사용하여 `examplebucket-1250000000` 버킷의 `exampleObject.txt` 객체와 `examplePrefix` 디렉터리의 모든 객체를 다운로드할 수 있는 권한을 사용자에게 부여하려면 액세스 정책은 다음과 같아야 합니다.

```
{
"version": "2.0",
"statement": [
{
"action":[
"name/cos:GetObject"
],
"effect": "allow",
"resource":[
"qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/exampleObject.txt",
"qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/examplePrefix/*"
]
}
]
}
```

사전 서명 최소 권한 가이드

사전 서명된 URL을 통해 임시 업로드 및 다운로드 작업을 구현할 수 있습니다. 또한 사전 서명된 URL을 누구에게나 전달할 수 있으며 유효한 사전 서명된 URL을 받는 사람은 누구나 객체를 업로드하거나 다운로드할 수 있습니다.

> 주의 :
> 임시 키와 영구 키를 모두 사용하여 사전 서명된 URL을 생성할 수 있지만, 최소 권한 가이드 임시 키 생성을 준수하여, 임시 키를 사용하여 사전 서명을 계산합니다. 보안 리스크를 방지하기 위해 과도한 권한을 가진 영구 키를 사용하지 마십시오.

권한 부여 예시

사전 서명된 URL을 사용하여 사용자에게 객체 다운로드 권한 부여

임시 키를 사용해 서명이 있는 다운로드 링크를 생성하고, 반환할 일부 공용 헤더(예: content-type, content-language)를 덮어쓰도록 설정합니다. 다음은 Java 예시 코드입니다.

```
// 획득한 임시 키(tmpSecretId, tmpSecretKey, sessionToken) 전송
String tmpSecretId = "SECRETID";
String tmpSecretKey = "SECRETKEY";
String sessionToken = "TOKEN";
COSCredentials cred = new BasicSessionCredentials(tmpSecretId, tmpSecretKey, sess
ionToken);
// bucket 리전 설정. COS 리전의 약칭은 https://cloud.tencent.com/document/product/43
6/6224를 참고하십시오.
// clientConfig에 region, https(기본값: http), 타임아웃, 프록시 등을 설정하는 set 메소
```

```
드가 포함되어 있습니다. 사용 시 소스 코드 또는 FAQ의 Java SDK 부분을 참고하십시오.
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
// https 프로토콜을 사용하는 URL을 생성할 경우, 이 행을 설정하길 권장합니다.
// clientConfig.setHttpProtocol(HttpProtocol.https);
// cos 클라이언트 생성.
COSClient cosClient = new COSClient(cred, clientConfig);
// 버킷의 이름 생성 형식: BucketName-APPID
String bucketName = "examplebucket-1250000000";
// 여기서 key는 객체 키로, 버킷 내 객체의 고유 식별자입니다.
String key = "exampleobject";
GeneratePresignedUrlRequest req =
new GeneratePresignedUrlRequest(bucketName, key, HttpMethodName.GET);
// 다운로드 시 반환하는 http 헤더 설정
ResponseHeaderOverrides responseHeaders = new ResponseHeaderOverrides();
String responseContentType = "image/x-icon";
String responseContentLanguage = "zh-CN";
// 반환 헤더에 포함되는 파일명 정보 설정
String responseContentDispositon = "filename=\"exampleobject\"";
String responseCacheControl = "no-cache";
String cacheExpireStr =
DateUtils.formatRFC822Date(new Date(System.currentTimeMillis() + 24L * 3600L * 10
00L));
responseHeaders.setContentType(responseContentType);
responseHeaders.setContentLanguage(responseContentLanguage);
responseHeaders.setContentDisposition(responseContentDispositon);
responseHeaders.setCacheControl(responseCacheControl);
responseHeaders.setExpires(cacheExpireStr);
req.setResponseHeaders(responseHeaders);
// 서명 만료 시간 설정(옵션). 설정하지 않을 경우 기본적으로 ClientConfig의 서명 만료 시간(1
시간)을 사용합니다.
// 본 예시에서는 30분 후 만료로 설정합니다.
Date expirationDate = new Date(System.currentTimeMillis() + 30L * 60L * 1000L);
req.setExpiration(expirationDate);
URL url = cosClient.generatePresignedUrl(req);
System.out.println(url.toString());
cosClient.shutdown();
```

사용자 정책 최소 권한 가이드

사용자 정책은 CAM 콘솔에 추가된 사용자 권한 정책으로, 사용자가 COS 리소스에 액세스할 수 있는 권한을 부여하는 데 사용됩니다. 사용자 액세스 정책 개요의 설정에 관한 자세한 설명은 액세스 정책 언어 개요 문서를 참고하십시오.

권한 부여 예시

계정에 지정된 객체에 액세스할 수 있는 권한 부여

UIN이 `100000000001` 인 계정에 `examplebucket-1250000000` 버킷의 `exampleObject.txt` 객체를 다운로드할 수 있는 권한을 부여하려면 액세스 정책은 다음과 같아야 합니다.

```
{
"version": "2.0",
"principal":{
"qcs":[
"qcs::cam::uin/100000000001:uin/100000000001"
]
},
"statement": [
{
"action":[
"name/cos:GetObject"
],
"effect": "allow",
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000.ap-guangzhou.myqcloud.com/exampleObject.txt"
]
}
]
}
```

서브 계정에 지정된 디렉터리에 대한 액세스 권한 부여

UIN이 `100000000011` 인 서브 계정(루트 계정 UIN: `100000000001`)에게 `examplebucket-1250000000` 버킷의 `examplePrefix` 디렉터리에 있는 객체를 다운로드할 수 있는 권한을 부여하려면 액세스 정책은 다음과 같아야 합니다.

```
{
"version": "2.0",
"principal":{
"qcs":[
"qcs::cam::uin/100000000001:uin/100000000011"
]
},
"statement": [
{
"action":[
"name/cos:GetObject"
],
"effect": "allow",
```

```
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000.ap-guangzhou.myqcloud.com/examplePrefix/*"
]
}
]
}
```

버킷 정책 최소 권한 가이드

버킷 정책은 버킷에서 설정한 액세스 정책으로, 지정 사용자가 버킷 및 버킷 내 리소스에 대해 지정된 작업을 할 수 있도록 허용합니다. 버킷 정책 설정은 버킷 정책 추가 문서를 참고하십시오.

권한 부여 예시

서브 계정에 지정된 객체에 대한 액세스 권한 부여

UIN이 `100000000011` 인 서브 계정(루트 계정 UIN: `100000000001`)에게 `examplebucket-1250000000` 버킷의 `exampleObject.txt` 객체와 `examplePrefix` 디렉터리의 모든 객체를 다운로드할 수 있는 권한을 부여하려면 액세스 정책은 다음과 같아야 합니다.

```
{
"Statement":[
{
"Action":[
"name/cos:GetObject"
],
"Effect": "allow",
"Principal":{
"qcs":[
"qcs::cam::uin/100000000001:uin/100000000011"
]
},
"Resource":[
"qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/exampleObject.txt",
"qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/examplePrefix/*"
]
}
],
"version": "2.0"
}
```

액세스 정책 평가 절차

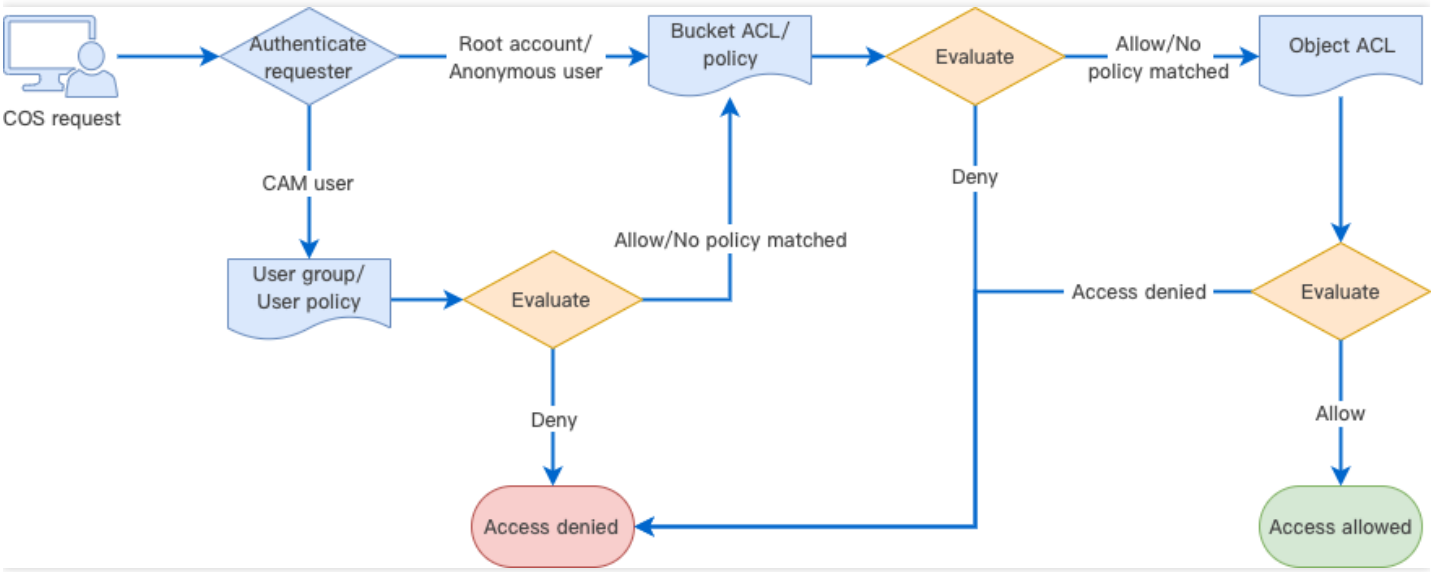
최종 업데이트 날짜: : 2022-08-16 15:24:13

COS 버킷과 버킷 리소스에 액세스할 경우 권한 부여 절차를 마쳐야만 액세스할 수 있습니다. Tencent Cloud 권한 시스템에서 리소스가 속한 루트 계정은 기본적으로 버킷과 버킷 리소스에 대한 모든 관리 권한을 가집니다. CAM 사용자(기타 루트 계정, 협업 파트너, 서브 계정) 그리고 익명 사용자 등의 기타 유형 사용자는 루트 계정으로부터 권한을 부여 받은 후 액세스 가능합니다.

계정의 액세스 정책은 사용자 그룹, 사용자 정책, 사용자 액세스 제어 리스트(ACL), 버킷 정책(Policy)등 다양한 정책 유형을 포함합니다. 액세스 정책 평가에서 중요한 요소는 다음과 같습니다.

1. 사용자 실명 인증: 사용자가 COS 리소스에 액세스할 때 다음과 같은 두 가지 상황이 있습니다.
   - 서명이 있는 경우 COS는 서명 요청을 하여 사용자의 계정 정보를 리졸브한 후 요청을 CAM에 포워딩하여 실명 인증을 진행합니다.
   - 서명이 없는 경우 익명 사용자로 분류되어 다음 링크의 인증으로 넘어갑니다.
2. 액세스 정책 구별: 액세스 정책은 사용자 그룹, 사용자, 버킷 등 다양한 유형의 정책을 포함하며 액세스 정책은 액세스 정책 순서에 따라 분류됩니다.
3. 정책 컨텍스트 정보: 리소스 액세스 요청을 프로세스 할 때 사용자 그룹 정책, 사용자 정책, 버킷 정책 등 여러 가지 정책 기록 권한의 세부 사항에 따라 종합적으로 판단하여 요청 통과를 결정합니다.

액세스 정책 평가 과정



Tencent Cloud COS가 요청을 받았을 때 먼저 요청자 자격을 확인하고 사용자 정책, 버킷 액세스 정책, 리소스를 기반으로 한 액세스 제어 리스트와 관련한 요청자의 권한 소유 여부를 인증한 후 요청을 확인합니다.

Tencent Cloud COS가 요청을 받으면 먼저 실명 인증을 진행하며 실명 인증 결과에 따라 요청자 자격을 분류하여 자격 별로 다양한 응답을 합니다.

1. 인증을 거친 Tencent Cloud 루트 계정: 루트 계정은 가지고 있는 리소스에 대한 모든 작업 권한을 가집니다. 그러나 권한을 가지고 있지 않은 리소스에 대해서는 리소스 권한 평가를 해야 합니다. 인증이 통과되면 리소스 액세스가 허용됩니다.
2. 인증을 거친 CAM 사용자(서브 계정 혹은 협업 파트너): 사용자 정책 평가, CAM 사용자는 반드시 상위 루트 계정이 권한을 부여해야 관련 액세스 허가를 받을 수 있습니다. CAM 사용자가 기타 루트 계정에 속한 리소스에 액세스 해야 할 경우 CAM 사용자가 속한 루트 계정의 리소스 권한 평가가 필요합니다. 인증이 통과되면 리소스 액세스가 허용됩니다.
3. 자격 특징이 없는 익명 사용자: 리소스 권한 평가 시, 버킷 액세스 정책 또는 버킷 및 객체의 액세스 제어 리스트의 권한을 평가합니다. 인증이 통과되면 리소스 액세스가 허용됩니다.
4. 상기 사용자 이외의 요청자는 액세스가 거부됩니다.

액세스 정책 평가 근거

Tencent Cloud 권한 시스템의 액세스 정책 평가 과정에서 모든 과정은 정책 컨텍스트 정보에 따라 권한 평가가 진행되며 다음 몇 가지 기본 원칙이 있습니다.

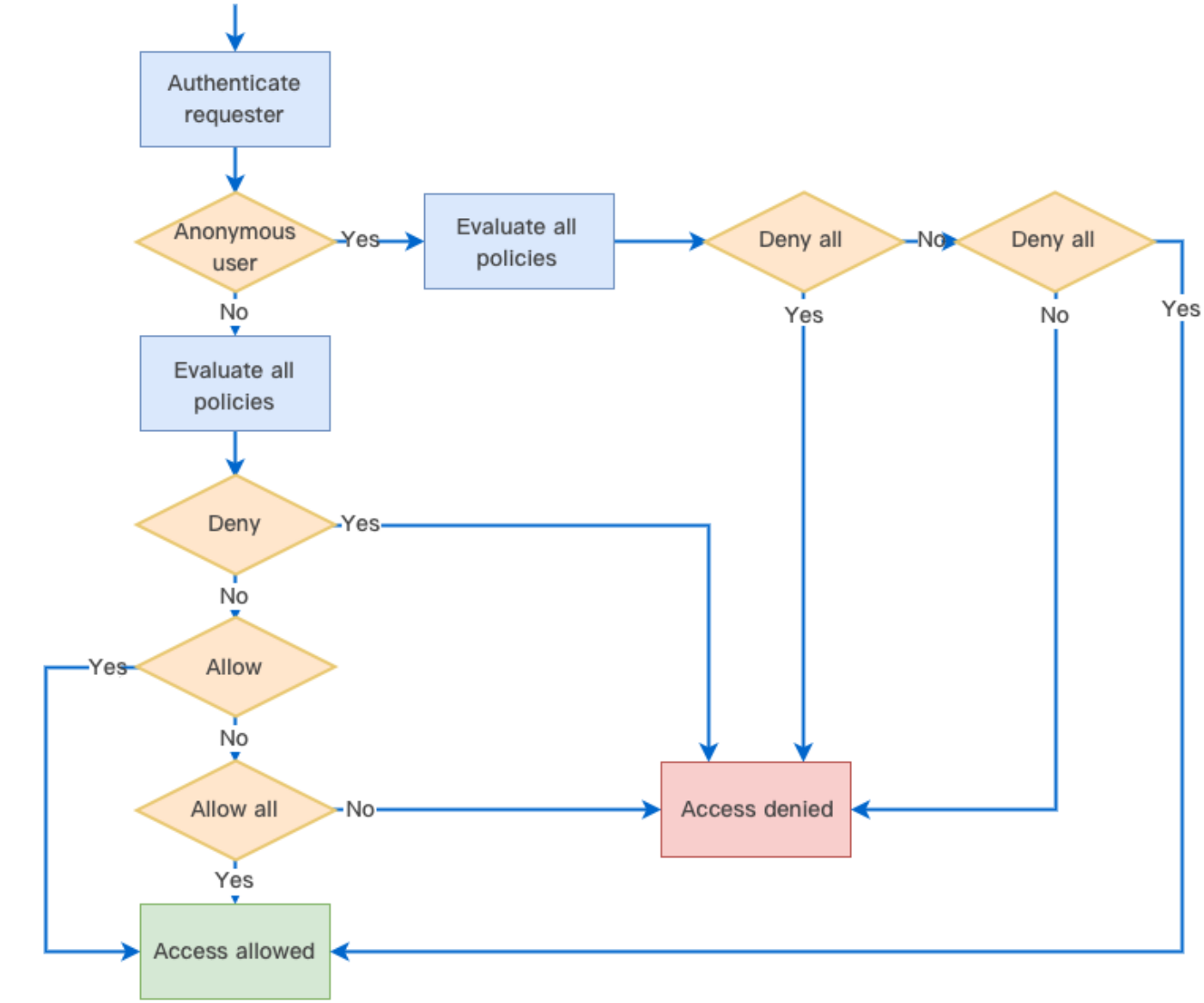
1. 모든 요청은 암묵적 거부(deny)가 기본 원칙입니다. 루트 계정은 계정에 속한 모든 리소스에 액세스할 수 있는 권한을 가집니다.
2. 사용자 그룹 정책, 사용자 정책, 버킷 정책 또는 버킷/객체 액세스 제어 리스트에서 명시적인 허용이 있는 경우 기본 값을 덮어씁니다.
3. 임의의 정책에서 명시적으로 거부하는 경우 임의의 허용을 덮어씁니다.
4. 적용 권한 범위는 자격에 따른 정책(사용자 그룹 정책, 사용자 정책)과 리소스에 따른 정책(버킷 정책이나 버킷/객체 액세스 제어 리스트) 결합에 따라 결정됩니다.

설명：

- 명시적 거부: 사용자 정책, 사용자 그룹 정책, 버킷 정책에서 특정 사용자에게 명확한 Deny 정책을 적용합니다. 예를 들어 루트 계정이 사용자 정책에서 `GET Object` 작업을 진행하는 서브 계정 UIN 100000000011에게 명시적인 Deny 정책을 적용했다면 그 서브 계정은 루트 계정의 객체 리소스를 다운로드할 수 없습니다.
- 명시적 허용: 사용자 정책, 사용자 그룹 정책, 버킷 정책, 버킷 ACL의 `grant-\*` 에서 특정 사용자에게 허용 정책을 적용합니다.
- 모든 사용자 거부: 버킷 정책에서 `Deny anyone` 을 명확히 지정하여 모든 사용자를 거부한 후 서명이 없는 임의의 요청은 거절하고 서명이 있는 요청은 자격 정책에 근거하여 인증을 진행합니다.

- 모든 사용자 허용: 버킷 정책에서 `Allow anyone` 을 지정하거나 버킷 ACL에서 `public-\*` 을 지정합니다.
- 적용 권한 범위는 자격 기반 정책과 리소스 기반 정책의 결합입니다. 한번 완전한 인증을 거칠 때 먼저 사용자의 자격을 리졸브하고 그 자격에 따라 액세스 권한을 가진 리소스가 있는지 여부를 가리는 권한 인증이 진행됩니다. 또한 리소스 기반 정책에 따라 익명 사용자로 보이는 사용자에 대해 권한 인증을 진행합니다. 두 번의 인증에서 한 번이라도 인증에 성공하면 액세스가 가능합니다.

액세스 정책 평가 근거는 아래 그림과 같이 보여집니다. 먼저 요청 중에 서명 소유 여부에 따라 익명 사용자인지 아닌지 평가합니다. 익명 사용자에 해당하는 경우 평가 정책을 모든 사용자 거부 또는 모든 사용자 허용 정책을 채택하며 그 판단에 따라 액세스 허용 및 거부가 이루어집니다. 사용자가 합법적인 CAM 사용자 또는 리소스를 가진 루트 계정이라면 평가 정책에서 명시적 거부, 명시적 허용 또는 모든 사용자 허용 정책을 채택하고 그 판단에 따라 액세스 허용 및 거부가 이루어집니다.



정책 컨텍스트 정보

정책 컨텍스트 정보는 정책 기록 권한 세부사항입니다. 최소 권한 원칙에서 사용자가 정책의 아래 정보를 명확히 지정해야 합니다.

- 위탁자(principal): 어떤 서브 계정(사용자 ID 작성 필요), 협업 파트너(사용자 ID 작성 필요), 익명 사용자 또는 사용자 그룹에게 권한 부여가 필요한지 명확히 지정할 필요가 있습니다. 임시 키로 액세스를 시도한다면 이 항목을 지정할 필요가 없습니다.
- 명령(statement): 다음 매개변수 중에 적합한 매개변수를 작성합니다.

- 효력(effect): 해당 정책에 대해 '허용'하는지 '명시적 거부'하는지 명확히 기술해야 합니다. allow와 deny 두 가지 상황이 포함됩니다.
- 작업(action): 해당 정책에 대해 허용 또는 거부 작업을 명확히 기술해야 합니다. 단일 API 작업이나 여러 API 작업의 결합 작업일 수 있습니다.
- 리소스(resource): 해당 정책이 권한을 부여한 구체적인 리소스를 명확히 기술해야 합니다. 리소스는 6단식 기술을 사용하며, 리소스 범위를 지정된 파일로 제한할 수 있습니다. 예를 들어 exampleobject.jpg 또는 examplePrefix/*와 같은 지정 디렉터리가 있습니다. 업무상 필요 외에는 모든 리소스에 액세스할 수 있는 권한 즉, 와일드카드*를 임의로 부여하지 마십시오.
- 조건(condition): 정책의 효력이 발생하는 규제 조건을 기술합니다. 조건에는 오퍼레이터, 작업 키와 작업 값 구성이 포함됩니다. 조건 값은 시간, IP 주소 등의 정보를 포함합니다.

정책을 작성할 경우 일정한 정책 구문에 따라 작성해야 하며 액세스 정책 언어 개요를 참조하십시오. 사용자 정책과 버킷 정책의 기입 예시는 각각 사용자 정책 언어 구조, 버킷 정책 예시를 참조하십시오.

액세스 정책 평가 예시

루트 계정 UIN 100000000001이 서브 계정 UIN 100000000011를 위해 사용자 사전 설정 정책을 연결하면, 서브 계정은 루트 계정의 리소스에 대해 읽기만 허용되며, 상세 내용은 아래와 같습니다. 해당 사용자 정책은 서브 계정이 `List` , `Get` , `Head` , `OptionsObject` 의 모든 작업을 실행하도록 허용합니다.

```
{
"version": "2.0",
"statement": [
{
"action": [
"cos:List*",
"cos:Get*",
"cos:Head*",
"cos:OptionsObject"
],
"resource": "*",
"effect": "allow"
}
]
}
```

또한 루트 계정은 개인 읽기/쓰기 버킷 `examplebucket-1250000000` 에 다음 버킷 정책을 추가합니다.

```
{
"Statement": [
{
```

```
"Principal": {
"qcs": [
"qcs::cam::anyone:anyone"
]
},
"Effect": "Deny",
"Action": [
"name/cos:GetObject"
],
"Resource": [
"qcs::cos:ap-guangzhou:uid/100000000011:examplebucket-1250000000/*"
]
}
],
"version": "2.0"
}
```

이 항목의 버킷 정책은 모든 사용자가 실행하는 객체 다운로드(`GetObject`) 작업을 명시적으로 거부합니다. 따라서 액세스 정책 평가 과정에서 다음과 같이 실행됩니다.

1. 서브 계정에 서명 매개변수 요청 `GetObject` 가 있는 경우, 해당 요청에 표시되는 사용자 자격에 부합하는 사용자 정책이 적용되어 액세스 정책 평가 인증이 통과됩니다.
2. 서브 계정에 서명 매개변수 요청 `GetObject` 가 없는 경우, 시스템에 의해 익명 요청으로 판단되며 버킷 정책에 의해 액세스가 **거부**됩니다.

권한 제어 방법 소개
버킷 정책

최종 업데이트 날짜: : 2022-04-22 14:42:48

버킷 정책은 설정된 버킷 및 버킷의 객체에 작용합니다. 버킷 정책을 통해 CAM 서브 계정, 다른 루트 계정 및 익명 사용자에 대한 버킷 및 버킷의 객체 작업 권한을 부여할 수 있습니다.

개요

> 주의 :
> Tencent Cloud 루트 계정은 그에 속한 리소스(버킷 포함)에 대해 가장 큰 권한을 가집니다. 버킷 정책이 거의 모든 작업을 제한하더라도, 루트 계정은 PUT Bucket Policy 작업 권한을 계속 가지고 있으므로, 해당 작업을 호출하여 버킷 정책을 점검하지 않도록 할 수 있습니다.

버킷 정책(Bucket Policy)은 JSON 언어를 사용하며 익명 자격이나 Tencent Cloud의 CAM 계정에 버킷, 버킷 작업, 객체 및 객체 작업에 대한 권한을 부여할 수 있습니다. Tencent Cloud COS의 버킷 정책은 버킷의 거의 모든 작업을 관리할 수 있습니다. 버킷 정책을 사용해 ACL로 나타낼 수 없는 액세스 정책을 관리하십시오.

적용 시나리오

> 주의 :
> 버킷 생성과 버킷 리스트 획득의 두 가지 서비스 레벨의 작업 권한은 CAM 콘솔을 통해 설정해야 합니다.

이 COS 버킷에 액세스할 수 있는 사용자에게 주의를 기울이는 경우 버킷 정책을 사용하는 것이 좋습니다. 버킷을 조회하고 버킷 정책을 확인하여 액세스할 수 있는 사용자를 확인할 수 있습니다. 권장 시나리오는 다음과 같습니다.

- 특정 버킷에만 권한을 부여하는 경우
- ACL에 비해 버킷 정책의 유연성이 더 높습니다.
- 사용자 정책 대비, 버킷 정책은 교차 계정 인증 및 익명 사용자 인증 지원

버킷 정책 구성

버킷 정책은 JSON 언어로 설명되어 있으며, 구문은 위탁자(principal), 효력(effect), 작업(action), 리소스(resource), 조건(condition) 등의 기본 요소를 포함하여 정책 언어 액세스의 통일된 규범을 따릅니다. 자세한 내용은 액세스 정책 언어 개요를 참고하십시오.
이 중 버킷 정책의 리소스 범위는 버킷 이내로 제한되며, 전체 버킷, 지정된 디렉터리, 지정된 객체에 대해 권한을 부여할 수 있습니다.

> 주의 :
> 버킷 정책을 추가할 때, 반드시 업무상 필요에 따라 최소 권한 원칙에 근거하여 인증해야 합니다. 다른 사용자에게 모든 리소스 `(resource:*)`, 또는 작업 `(action:*)` 에 대한 모든 권한을 부여하면 권한 범위가 너무 광범위해 데이터 보안에 리스크가 발생할 수 있습니다.

콘솔 설정 예시

> 설명 :
> - COS 콘솔을 이용해 버킷 정책을 설정할 때, 사용자에게 버킷 소유와 관련된 권한(예: 버킷 태그 얻기, 버킷 권한 나열)을 부여해야 합니다.
> - 버킷 정책의 크기는 20KB로 제한됩니다.

예시: 서브 계정에 버킷의 특정 디렉터리에 대한 모든 권한을 부여합니다. 설정 정보는 다음과 같습니다.

설정 항목	설정값
효과	허용
위탁자	서브 계정의 UIN이며, 해당 서브 계정은 반드시 현재 루트 계정의 서브 계정이어야 합니다(예: 100000000011)
리소스	특정 디렉터리의 접두사(예: `folder/sub-folder/*`)
작업	모든 작업
조건	없음

콘솔은 [그래픽 설정](#그래픽 설정) 및 [정책 설정](#정책 설정) 두 가지 방식의 버킷 정책 추가, 관리를 지원합니다.

그래픽 설정

타깃 버킷의 **권한 관리**로 이동하여 **Policy 권한 설정** > **그래픽 설정**을 선택하고 **정책 추가**를 클릭하여 팝업 창에서 정책을 설정합니다.

1단계: 템플릿 선택(옵션)

권한이 부여된 사용자와 리소스 범위의 다양한 조합을 선택함으로써 COS는 버킷 정책의 빠른 구성에 도움이 되는 다양한 정책 템플릿을 제공합니다. 템플릿이 요구 사항을 충족하지 않는 경우 이 단계를 건너뛰거나 2단계: 정책 설정에서 권한 부여 작업을 추가 또는 삭제할 수 있습니다.

펼치기

인증된 사용자

展开&收起

- **모든 사용자(익명 액세스 가능)**: 익명 사용자에 대한 작업 권한을 개방하려면 이 항목을 선택합니다. 다음 단계에서 정책을 설정할 때 모든 사용자가 자동으로 추가되며 `*` 로 표시됩니다.
- **지정된 사용자**: 지정된 서브 계정, 루트 계정 또는 클라우드 서비스에 운영 권한을 개방하려면 이 항목을 선택합니다. 다음 단계에서 정책 설정 시 특정 계정 UIN을 추가로 지정해야 합니다.

리소스 범위

展开&收起

- **전체 버킷**: 버킷 설정 항목과 관련된 권한을 부여하거나 리소스 범위를 전체 버킷으로 지정하려면 이 항목을 선택하면 2단계 정책 설정 시 전체 버킷이 자동으로 리소스로 추가됩니다.
- **디렉터리 지정**: 리소스 범위를 지정된 폴더로 제한하려는 경우 이 항목을 선택합니다. 2단계 정책 설정 시 특정 디렉터리를 추가로 지정해야 합니다.

템플릿

展开&收起

권한을 부여할 작업 그룹입니다. 선택한 권한 부여 사용자 및 리소스 범위에 따라 COS는 권장 정책 템플릿을 제공합니다. 템플릿이 요구 사항을 충족하지 않는 경우 이 단계를 건너뛰거나 다음 단계 '정책 설정'에서 권한 부여 작업을 추가 또는 삭제할 수 있습니다.

- **사용자 정의 정책(사전 설정 미제공)**: 템플릿을 사용할 필요가 없는 경우 이 옵션을 선택하고 2단계 '정책 설정'에서 필요에 따라 정책을 추가할 수 있습니다.
- **기타 템플릿**: 인증된 사용자 및 리소스 범위의 다양한 조합 선택에 따라 COS는 다양한 정책 템플릿을 제공합니다. 해당 템플릿을 선택한 후, 2단계 정책 설정에서 COS가 자동으로 해당 작업을 추가합니다.

템플릿 설명은 아래 표를 참고하십시오.

인증된 사용자	리소스 범위	정책 템플릿	설명

모든 조합		사용자 정의 정책	인증된 사용자 및 리소스 범위의 조합에 대해 이 템플릿을 선택하면 사전 설정된 정책이 제공되지 않습니다. 2단계 정책 설정에서 정책을 직접 추가할 수 있습니다.
모든 사용자(익명 액세스 가능)	전체 버킷	읽기 전용 객체(객체 리스트 제외)	COS는 익명 사용자에 대해 파일 읽기(예: 다운로드) 및 파일 쓰기(예: 업로드 및 수정) 권장 템플릿을 제공합니다. COS 권장 템플릿에는 데이터 보안을 위해 버킷의 모든 객체 나열, 읽기/쓰기 권한, 버킷 설정 등 민감한 권한이 포함되어 있지 않습니다. 필요한 경우 후속 단계에서 작업 권한을 추가 또는 삭제할 수 있습니다.
		객체 읽기 및 쓰기(객체 리스트 나열 제외)	
	지정된 디렉터리	읽기 전용 객체(객체 리스트 제외)	
		객체 읽기 및 쓰기(객체 리스트 나열 제외)	
지정된 사용자	전체 버킷	읽기 전용 객체(객체 리스트 제외)	COS는 지정된 사용자와 전체 버킷의 조합에 대해 가장 많은 권장 템플릿을 제공합니다. 파일 읽기 및 쓰기, 파일 나열 외에도 COS에는 신뢰할 수 있는 사용자에게 적합한 다음과 같은 민감 권한 템플릿이 포함되어 있습니다. • 버킷 및 객체 ACL 읽기 및 쓰기: 버킷 ACL 및 객체 ACL 가져오기, 수정. GetObjectACL, PutObjectACL, GetBucketACL, PutBucketACL 포함. • 버킷 일반 설정 항목: 버킷 태그, 교차 출처, Origin-pull 및 기타 민감하지 않은 권한. • 버킷 민감 설정 항목: 버킷 정책, 버킷 ACL, 버킷 삭제와 같은 민감한 권한이 포함되므로 신중히 사용해야 합니다.
		읽기 전용 객체(객체 리스트 나열 포함)	
		객체 읽기 및 쓰기(객체 리스트 나열 제외)	
		객체 읽기 및 쓰기(객체 리스트 나열 포함)	
		버킷 및 객체 ACL 읽기 및 쓰기	

		버킷에 대한 일반 설정 항목	
		버킷에 대한 민감 설정 항목	
	지정된 디렉터리	읽기 전용 객체(객체 리스트 제외)	COS는 지정된 사용자와 지정된 디렉터리의 조합에 대해 파일 읽기(예: 다운로드) 및 파일 쓰기(예: 업로드, 수정) 외에도 , 객체 리스트 나열 권한을 포함하는 권장 템플릿을 제공합니다. 지정된 사용자에 대해 지정된 폴더의 읽기, 쓰기 및 파일 나열 권한을 개방하려면 이 조합을 권장합니다. 필요에 따라 후속 단계에서 작업 권한을 추가하거나 삭제할 수 있습니다.
		읽기 전용 객체(객체 리스트 나열 포함)	
		객체 읽기 및 쓰기(객체 리스트 나열 제외)	
		객체 읽기 및 쓰기(객체 리스트 나열 포함)	

2단계: 정책 설정

1단계에서 선택한 인증된 사용자, 지정된 디렉터리 및 템플릿의 조합에 대해, COS는 해당 작업, 인증된 사용자, 리소스 등을 정책 설정에 자동으로 추가합니다. 지정된 사용자 및 지정된 디렉터리를 선택한 경우, 정책 설정 시 특정 사용자 UIN 및 디렉터리를 지정해야 합니다.

> 설명：
> 디렉터리 권한 부여를 위해서는 입력 리소스 경로 뒤에 `/*` 가 와야 합니다. 예를 들어, 디렉터리 test에 권한을 부여하려면 `test/*` 를 입력합니다.

COS에서 제공한 권장 템플릿이 요구 사항에 맞지 않을 경우 이 단계에서 인증된 사용자, 리소스 및 작업 추가 또는 삭제 등 정책 내용을 수정할 수 있습니다. 다음 이미지와 같습니다.

다음은 설정 항목에 관한 설명입니다.

- **유효성**: 정책 구문의 **허용** 및 **거부**에 해당하는 'allow' 또는 'deny' 선택을 지원합니다.
- **사용자**: 모든 사용자(`*`), 루트 계정, 서브 계정 및 클라우드 서비스 등 인증된 사용자 추가 및 삭제를 지원합니다.
- **리소스**: 전체 버킷 또는 지정된 디렉터리 리소스 추가를 지원합니다.
- **작업**: 인증해야 하는 작업을 추가 또는 삭제합니다.
- **조건**: 사용자의 IP 액세스 제한 등 권한 부여 시 조건을 지정합니다.

정책 구문

그래픽 설정을 사용하는 것 외에도 버킷 정책에 익숙한 사용자는 JSON 언어를 사용하여 타깃 버킷의 **권한 관리** > **Policy 권한 설정** > **정책 설정**에서 정책을 직접 작성할 수 있습니다.

버킷 정책을 작성 후 API 또는 SDK를 통해 버킷 정책을 추가할 수도 있습니다. 아래 이미지와 같습니다.

JSON 정책 예시

다음 정책 예시는 다음과 같이 설명됩니다. 기본 계정 ID 100000000001(APPID는 1250000000) 하위 서브 계정 ID 100000000011에 속한 베이징 리전의 examplebucket-bj 버킷의 'folder/sub-folder' 디렉터리 중의 객체에 대한 모든 작업에 권한을 부여합니다.

```
{
"Statement": [
{
"Principal": {
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000011"
]
},
"Effect": "Allow",
"Action": [
"name/cos:*"
],
"Resource": [
"qcs::cos:ap-beijing:uid/1250000000:examplebucket-bj-1250000000/folder/sub-folde
r/*"
]
}
],
"version": "2.0"
}
```

작업 방식

COS는 콘솔, API, SDK 등을 사용한 버킷 정책 추가를 지원합니다. 콘솔은 그래픽 설정과 일반 권한 부여 템플릿을 지원하므로 정책 언어에 익숙하지 않은 사용자가 정책을 빠르게 추가할 수 있습니다.

작업 방식		설명
콘솔		Web 페이지, 직관적이고 편리함
API		RESTful API, 직접 COS 요청
SDK	JavaScript	풍부한 SDK demo로, 다양한 개발 언어를 지원합니다.
	Node.js	
	미니프로그램	

더 많은 버킷 정책 예시

- 버킷 정책(Policy)을 통한 권한 부여 사례
- 기타 루트 계정의 서브 계정에 관련 버킷 실행 권한 부여

> 주의 :
> 버킷 정책을 추가할 때, 반드시 업무상 필요에 따라 최소 권한 원칙에 근거하여 권한을 부여해야 합니다. 다른 사용자에게 모든 리소스 `(resource:*)` , 또는 작업 `(action:*)` 에 대한 모든 권한을 부여하면 권한 범위가 너무 광범위해 데이터 보안에 리스크가 발생할 수 있습니다.

다음은 서브넷, 위탁자 및 VPC ID를 제한하는 버킷 정책의 예시입니다.

예시1

서브넷 10.1.1.0/24 IP 대역 및 vpcid의 aqp5jrc1에 대한 요청을 제한합니다. 구문 예시는 다음과 같습니다.

```
{
"Statement": [
{
"Action": [
"name/cos:*"
],
"Condition": {
"ip_equal": {
"qcs:ip": [
"10.1.1.0/24"
```

```
]
},
"string_equal": {
"vpc:requester_vpc": [
"vpc-aqp5jrc1"
]
}
},
"Effect": "deny",
"Principal": {
"qcs": [
"qcs::cam::anyone:anyone"
]
},
"Resource": [
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
]
}
],
"version": "2.0"
}
```

예시2

vpcid의 aqp5jrc1 및 특정 위탁자와 특정 버킷에 대한 요청을 제한합니다. 구문 예시는 다음과 같습니다.

```
{
"Statement": [
{
"Action": [
"name/cos:*"
],
"Condition": {
"string_equal": {
"vpc:requester_vpc": [
"vpc-aqp5jrc1"
]
}
},
"Effect": "allow",
"Principal": {
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000002"
]
},
"Resource": [
```

```
"qcs::cos:ap-beijing:uid/1250000000:examplebucket-1250000000/*"
]
}
],
"version": "2.0"
}
```

사용자 정책

최종 업데이트 날짜: : 2022-12-01 14:19:22

Tencent Cloud 루트 계정은 CAM(Cloud Access Management) 콘솔에서 CAM 사용자를 생성하고 정책을 연결하여 CAM 사용자에게 Tencent Cloud 리소스 사용 권한을 부여할 수 있습니다.

개요

사용자는 CAM에서 루트 계정에 속한 다양한 유형의 사용자에게 여러 가지 권한을 부여할 수 있습니다. 이러한 권한은 액세스 정책 언어로 설명되고 사용자를 시작점으로 권한이 부여되므로 **사용자 정책**이라고 합니다.

사용자 정책과 버킷 정책의 차이점

사용자 정책과 버킷 정책의 가장 큰 차이점은 사용자 정책은 효력(Effect), 작업(Action), 리소스(Resource), 조건(Conditon, 옵션)만 설명하며 자격(Principal)은 설명하지 않는다는 점입니다. 이 때문에 사용자 정책 작성이 완료되면 다시 서브 계정, 사용자 그룹 또는 역할 관련 작업을 실행해야 합니다. 따라서 사용자 정책의 사용 방법은 다음과 같습니다.

- 사용자 정책 작성이 완료된 후, 서브 계정, 사용자 그룹 또는 역할에 대한 관련 작업을 수행합니다.
- 사용자 정책은 **익명 사용자에게 작업 및 리소스 권한 부여를 지원하지 않습니다**.

사전 설정 정책 및 사용자 정의 정책

사용자 정책에는 사전 설정 정책 및 사용자 정의 정책의 두 가지 유형이 있으며, 연결 인증을 위한 사전 설정 정책을 사용하거나 직접 사용자 정책 작성 후 연결 인증을 수행할 수도 있습니다. 자세한 내용은 CAM의 인증 가이드를 참고하십시오.

적용 시나리오

사용자가 무엇을 할 수 있는지 궁금하거나 사용자 정책을 추천할 때, CAM 사용자를 찾고 그들이 속한 사용자 그룹의 권한을 확인하여 사용자가 할 수 있는 일을 알아보십시오. 권장 시나리오는 다음과 같습니다.

- 버킷 생성(PutBucket) 및 버킷 나열(GetService)과 같은 COS(Cloud Object Storage) 서비스 수준 권한을 설정합니다.
- 루트 계정의 모든 COS 버킷 및 객체를 사용해야 합니다.
- 루트 계정의 다수의 CAM 사용자에게 동일한 권한을 부여합니다.

사용자 정책 구문

정책 구문

버킷 정책과 마찬가지로 사용자 정책은 JSON 언어로 설명되며 액세스 정책 언어의 통합 표준(위탁자, 효과, 작업, 리소스, 조건 등)을 따릅니다. 그러나 사용자 정책은 사용자/사용자 그룹에 직접 연결되므로 사용자 정책은 위탁자(Principal)를 입력할 필요가 없습니다.

다음 표에서는 사용자 정책과 버킷 정책 간의 차이점 비교입니다.

요소	사용자 정책	버킷 정책
위탁자	**입력하지 않음**	필수 입력
효과	필수 입력	필수 입력
작업	필수 입력	필수 입력
리소스	필수 입력	**버킷 리소스**
조건	옵션	옵션

정책 예시

다음은 다음은 일반적인 사용자 정책의 예시이며, 정책의 의미는 광저우에 위치한 버킷 examplebucket-1250000000의 모든 COS 작업에 권한 위임을 한 정책입니다. 정책을 저장한 후 CAM의 서브 계정, 사용자 그룹 또는 적용 가능한 역할과 연결해야 합니다.

```
{
"Statement": [
{
"Effect": "Allow",
"Action": ["cos:*"],
"Resource": [
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*",
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/"
]
}
],
"Version": "2.0"
}
```

사용자 정책을 통해 서브 계정에 COS 액세스 인증

전제 조건

CAM 서브 계정이 생성되어야 합니다. 생성 방법은 서브 계정 생성을 참고하십시오.

설정 단계

CAM은 사전 설정 정책 및 사용자 정의 정책을 제공합니다. 사전 설정 정책은 CAM에서 제공하는 시스템 사전 설정 정책으로, COS 관련 정책은 [사전 설정 정책](#사전 설정 정책)을 참고하십시오. 사용자 정의 정책은 사용자 정의 리소스, 작업 등 요소를 지원하여 보다 효율적으로 만듭니다. 서브 계정을 인증하는 사용자 정의 정책 생성 방법은 다음과 같습니다.

1. CAM 콘솔에 로그인합니다.

2. **정책 > 사용자 정의 정책 생성 > 정책 구문 생성**을 선택하여 정책 생성 페이지로 이동합니다.

3. **빈 템플릿**을 선택하여 실제 필요에 따라 인증 정책을 사용자 정의하거나 COS와 연결된 **시스템 템플릿**을 선택할 수 있습니다. 다음은 **빈 템플릿**을 선택하는 예시입니다.

4. **빈 템플릿**을 선택하여 정책 구문을 입력합니다.다음과 같은 기본 요소가 포함되어야 합니다.

- **resource: 권한 부여 리소스**.
  - 모든 리소스(`"*"`)
  - 버킷 지정(`"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"`)
  - 버킷의 지정된 디렉터리 또는 객체(`"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/test/*"`)
- **action: 권한 부여 작업**.
- **effect: 효력**. `"allow"` (허용) 또는 `"deny"` (거절)을 선택합니다.
- **condition: 적용 조건**. 옵션 항목.

  COS는 사용자 정책의 예시를 제공하고 있으며, 다음 문서를 참고하여 정책 콘텐츠를 직접 복사하여 **정책 콘텐츠** 편집 상자에 붙여넣을 수 있습니다. 입력이 올바른지 확인한 후 **완료**를 클릭하십시오.

- 서브 계정 COS 액세스 인증
- COS API 인증 정책 사용 가이드

5. 생성이 완료되면 CAM 콘솔의 **정책 > 사용자 정의 정책**에서 생성된 사용자 정의 정책을 확인하고 정책을 서브 계정과 연결할 수 있습니다.

6. 서브 계정을 선택하고 **확인**을 클릭해 권한을 부여하면, 서브 계정을 이용해 한정된 COS 리소스에 액세스할 수 있습니다.

사전 설정 정책

1. CAM은 몇 가지 사전 설정 정책을 제공합니다. CAM 콘솔의 **정책 > 사전 설정 정책**에서 'COS'를 검색하여 필터링할 수 있습니다.

2. 정책 이름을 클릭하고 **정책 구문 > JSON**으로 이동하여 특정 정책 콘텐츠를 확인합니다. 사전 설정된 정책의 리소스(`resource`)는 COS의 모든 리소스(`"*"`)로 설정되며 수정은 지원되지 않습니다. 일부 COS 버킷 및 객체에 인증을 부여해야 하는 경우 JSON 사전 설정 정책을 복사하여 [사용자 정의 정책](#사용자 정의 정책)을 생성할 수 있습니다.

표1과 표2는 CAM에서 제공하는 COS 관련 사전 설정 정책 및 설명을 나열합니다.

표1: COS 사전 설정 정책

사전 설정 정책	설명	JSON 정책

QcloudCOSBucketConfigRead	이 권한이 있는 사용자는 COS 버킷 설정 읽기 가능	<pre>{
		"version": "2.0",
		"statement": [
		{
		"effect": "allow",
		"action": [
		"cos:GetBucket*",
		"cos:HeadBucket"
		],
		"resource": "*"
		}
		]
		}</pre>

QcloudCOSBucketConfigWrite	이 권한이 있는 사용자는 COS 버킷 설정 수정 가능	<pre>{
		"version": "2.0",
		"statement": [</pre>

		<pre>{
		"effect": "allow",
		"action": [
		"cos:PutBucket*"
		],
		"resource": "*"
		}
		]
		}</pre>

QcloudCOS Data FullControl	COS 버킷의 데이터 읽기, 쓰기, 삭제 및 나열할 수 있는 액세스 권한 포함	<pre>{
		"version": "2.0",
		"statement": [
		{
		"effect": "allow",
		"action": [
		"cos:GetService",
		"cos:GetBucket",
		"cos:ListMultipartUploads",
		"cos:GetObject*",
		"cos:HeadObject",
		"cos:GetBucketObjectVersions",
		"cos:OptionsObject",
		"cos:ListParts",
		"cos:DeleteObject",
		"cos:PostObject",
		"cos:PostObjectRestore",
		"cos:PutObject*",
		"cos:InitiateMultipartUpload",
		"cos:UploadPart",
		"cos:UploadPartCopy",
		"cos:CompleteMultipartUpload",
		"cos:AbortMultipartUpload",
		"cos:DeleteMultipleObjects",
		"cos:AppendObject"
		],</pre>

		"resource": "*" }] }

표2: COS 작업과 사전 설정 정책 간의 관계

설명	작업	QcloudCOS Bucket ConfigRead	QcloudCOS Bucket ConfigWrite	QcloudCOS Data FullControl	QcloudCOS Data ReadOnly	Qclou Data WriteC
버킷 나열	GetService	✕	✕	✓	✕	✕
버킷 생성	PutBucket	✕	✓	✕	✕	✕
버킷 삭제	DeleteBucket	✕	✕	✕	✕	✕
기본 버킷 정보 가져오기	HeadBucket	✓	✕	✕	✕	✕
버킷 설정 항목 가져오기	GetBucket*	✓	✕	✕	✕	✕
버킷 설정 항목 수정	GetBucket*	✕	✓	✕	✕	✕
버킷 액세스 권한 가져오기	GetBucketAcl	✓	✕	✕	✕	✕

설명	작업	QcloudCOS Bucket ConfigRead	QcloudCOS Bucket ConfigWrite	QcloudCOS Data FullControl	QcloudCOS Data ReadOnly	Qclou Data WriteC
버킷 액세스 권한 수정	PutBucketAcl	✕	✓	✕	✕	✕
버킷의 객체 나열	GetBucket	✓	✕	✓	✕	✕
버킷의 모든 버전 객체 나열	GetBucketObjectVersions	✓	✕	✓	✕	✕
객체 업로드	PutObject	✕	✕	✓	✕	✓
멀티파트 업로드	ListParts InitiateMultipartUpload UploadPart UploadPartCopy CompleteMultipartUpload AbortMultipartUpload ListMultipartUploads	✕	✕	✓	✕	✓
객체 추가	AppendObject	✕	✕	✓	✕	✕
객체 다운로드	GetObject	✕	✕	✓	✓	✕
객체 메타데이터 조회	HeadObject	✕	✕	✓	✓	✕

설명	작업	QcloudCOS Bucket ConfigRead	QcloudCOS Bucket ConfigWrite	QcloudCOS Data FullControl	QcloudCOS Data ReadOnly	Qclou Data WriteC
CORS 사전 인증	OptionsObject	✘	✘	✔	✔	✘

더 많은 사용자 정책 예시

- COS 액세스 서브 계정 인증
- COS API 인증 정책 사용 가이드

ACL

최종 업데이트 날짜: : 2022-12-29 11:26:22

기본 개념

액세스 제어 리스트(ACL)는 XML 언어를 사용하며, 리소스와 관련된 권한 부여자 및 권한 수여자 리스트입니다. 각각의 버킷과 객체에는 모두 이와 관련된 ACL이 있으며, 익명 사용자나 다른 Tencent Cloud의 루트 계정에 기본적인 읽기 및 쓰기 권한을 부여합니다.

주의 :
리소스와 관련된 ACL 관리에는 다음과 같은 제한이 있습니다.

- 리소스 소유자는 항상 리소스에 대해 FULL_CONTROL 권한을 가지며, 권한을 철회 또는 수정할 수 없습니다.
- 익명 사용자는 리소스 소유자가 될 수 없으며, 이때 객체 리소스의 소유자는 버킷의 생성자(Tencent Cloud 루트 계정)에 속합니다.
- Tencent Cloud CAM(Cloud Access Management) 루트 계정 또는 사전 설정 사용자 그룹에만 권한을 부여할 수 있으며 사용자 정의 사용자 그룹에는 권한을 부여할 수 없으므로 서브 계정에게 권한을 부여하지 않는 것이 좋습니다.
- 권한에 대한 추가 조건은 지원하지 않습니다.
- 거부 표시 권한은 지원하지 않습니다.
- 하나의 리소스는 최대 100개의 ACL 정책을 가질 수 있습니다.

적용 시나리오

주의 :
공개 익명 사용자 액세스(공개 읽기)는 매우 위험한 작업으로 트래픽 도용의 리스크가 있으므로 공개 읽기를 사용해야 하는 경우 보안을 위해 링크 도용 방지 설정을 할 수 있습니다.

간단한 액세스 권한을 설정하거나 또는 버킷 및 객체에 대한 익명 액세스를 개방해야 하는 경우 ACL을 선택하십시오. 그러나 더 많은 경우에는 더 유연한 버킷 정책 또는 사용자 정책을 먼저 사용하는 것이 좋습니다. ACL에 적용 가능한 시나리오는 다음과 같습니다.

- 단순 액세스 권한만 설정합니다.
- 콘솔에서 액세스 권한을 빠르게 설정합니다.
- 모든 익명의 인터넷 사용자에게 객체, 디렉터리 또는 버킷을 개방하며 ACL 작업이 더 편리합니다.

ACL의 요소

자격 Grantee

CAM의 루트 계정 또는 사전 설정된 CAM 사용자 그룹은 권한을 수여할 수 있습니다.

> 주의 :
>
> - 기타 Tencent Cloud 루트 계정에 액세스 권한을 부여했을 때, 이 권한을 받은 루트 계정은 해당 루트 계정에 속해 있는 서브 계정, 사용자 그룹 또는 역할에 액세스 권한을 부여할 수 있습니다.
> - COS(Cloud Object Storage)는 익명 사용자 또는 CAM 사용 그룹에 WRITE, WRITE_ACP 또는 FULL_CONTROL 권한 부여를 권장하지 않습니다. 권한 부여를 허용하면 사용자 그룹이 귀하의 리소스를 업로드/다운로드/삭제할 수 있고 이로 인해 데이터 손실, 비용 차감 등 리스크가 발생할 수 있습니다.

버킷 또는 객체의 ACL에서 권한을 수여할 수 있는 자격은 다음을 포함합니다.

- 계정 간: 루트 계정의 ID를 사용하고, **계정 센터**의 계정 정보를 통해 계정 ID를 받으십시오. (예시: 100000000001)
- 사전 설정 사용자 그룹: URI 태그를 사용해 사전 설정된 사용자 그룹을 표기하십시오. 지원되는 사용자 그룹은 다음을 포함합니다.
  - 익명 사용자 그룹 - `http://cam.qcloud.com/groups/global/AllUsers` 해당 그룹은 요청에 서명을 했든지 안 했든지, 누구나 권한 부여 없이 리소스에 액세스할 수 있다는 의미입니다.
  - 인증 사용자 그룹 - `http://cam.qcloud.com/groups/global/AuthenticatedUsers` 해당 그룹은 Tencent Cloud CAM 계정의 인증을 거친 사용자는 모두 리소스에 액세스할 수 있다는 의미입니다.

작업 Permission

Tencent Cloud COS가 리소스 ACL에서 지원하는 작업은 사실상 일련의 작업 그룹으로, 버킷 및 객체 ACL에는 각각 다른 의미를 갖습니다.

버킷의 작업

아래 표는 버킷 ACL에서 설정을 지원하는 작업 리스트입니다.

작업 그룹	설명	허용된 동작
READ	객체 나열	HeadBucket, GetBucketObjectVersions, ListMultipartUploads

작업 그룹	설명	허용된 동작
WRITE	객체 업로드, 덮어쓰기 및 삭제	PutObject, PutObjectCopy, PostObject, InitiateMultipartUpload, UploadPart, UploadPartCopy, CompleteMultipartUpload, DeleteObject
READ_ACP	버킷의 ACL 읽기	GetBucketAcl
WRITE_ACP	버킷의 ACL 쓰기	PutBucketAcl
FULL_CONTROL	이상 네 가지 권한의 집합	이상 모든 동작의 집합

주의：
버킷의 WRITE, WRITE_ACP 또는 FULL_CONTROL 권한 부여에 신중하시기 바랍니다. 버킷에 부여된 WRITE 권한은 권한 수여자 모든 객체를 덮어쓰기 또는 삭제할 수 있도록 허용합니다.

객체의 작업

아래 표는 객체 ACL에서 설정을 지원하는 작업 리스트입니다.

작업 그룹	설명	허용된 동작
READ	객체 읽기	GetObject, GetObjectVersion, HeadObject
READ_ACP	객체의 ACL 읽기	GetObjectAcl, GetObjectVersionAcl
WRITE_ACP	객체의 ACL 쓰기	PutObjectAcl, PutObjectVersionAcl
FULL_CONTROL	이상 세 가지 권한의 집합	이상 모든 동작의 집합

설명：
객체는 WRITE 작업 그룹 권한 부여를 지원하지 않습니다.

사전 설정한 ACL

COS는 일련의 사전 설정된 ACL에 대한 권한 부여를 지원하며 간단한 권한을 쉽게 설명할 수 있습니다. ACL 사전 설정을 사용할 때, PUT Bucket/Object 또는 PUT Bucket/Object acl에서 x-cos-acl 헤더를 가져옴과 동시에 필요한 권한을 설명합니다. 만약 동시에 본문 요청 중 XML의 설명 콘텐츠를 가져왔을 경우, 헤더 설명을 먼저 선택하고 본문 요청 중의 XML 설명은 무시합니다.

버킷의 사전 설정 ACL

사전 설정 명칭	설명
private	생성자(루트 계정)는 FULL_CONTROL 권한을 가지며, 다른 사람은 권한이 없습니다.(기본값)
public-read	생성자는 FULL_CONTROL 권한을 가지며, 익명 사용자 그룹은 READ 권한을 갖습니다.
public-read-write	생성자와 익명 사용자 그룹이 모두 FULL_CONTROL 권한을 가지며, 이 권한을 부여하는 것은 권장하지 않습니다.
authenticated-read	생성자는 FULL_CONTROL 권한을 가지며, 인증 사용자 그룹은 READ 권한을 갖습니다.

객체의 사전 설정 ACL

사전 설정 명칭	설명
default	설명이 비어 있을 때, 각 레벨 디렉터리의 명시적 설정 및 버킷의 설정에 따라 요청이 허용됐는지 확인합니다.(기본값)
private	생성자(루트 계정)는 FULL_CONTROL 권한을 가지며, 다른 사람은 권한이 없습니다.
public-read	생성자는 FULL_CONTROL 권한을 가지며, 익명 사용자 그룹은 READ 권한을 갖습니다.
authenticated-read	생성자는 FULL_CONTROL 권한을 가지며, 인증 사용자 그룹은 READ 권한을 갖습니다.
bucket-owner-read	생성자는 FULL_CONTROL 권한을 가지며, 버킷 소유자는 READ 권한을 갖습니다.
bucket-owner-full-control	생성자와 버킷 소유자가 FULL_CONTROL 권한을 갖습니다.

설명：
객체는 public-read-write 권한 부여를 지원하지 않습니다.

예시

버킷의 ACL

버킷 생성 시 COS는 기본 ACL을 생성하여, 리소스 소유자가 리소스에 대한 전체 제어 권한을 부여 받도록 합니다. 예시는 다음과 같습니다.

```
<AccessControlPolicy>
<Owner>
<ID>Owner-Cononical-CAM-User-Id</ID>
</Owner>
<AccessControlList>
<Grant>
<Grantee>
<ID>Owner-Cononical-CAM-User-Id</ID>
</Grantee>
<Permission>FULL_CONTROL</Permission>
</Grant>
</AccessControlList>
</AccessControlPolicy>
```

객체의 ACL

객체 생성 시, COS는 기본적으로 ACL을 생성하지 않으며, 이때 객체의 소유자는 버킷 소유자입니다. 객체 상속 버킷의 권한은 버킷의 액세스 권한과 일치합니다. 객체는 기본적으로 ACL을 가지지 않으므로 버킷 정책(Bucket Policy) 중 방문자와 그의 동작의 정의에 따라 요청이 허용 여부를 판단합니다. 자세한 내용은 액세스 정책 언어 개요 문서를 참고하십시오.

객체에 기타 액세스 권한을 부여해야 할 경우, 이를 기초로 더 많은 ACL을 추가해 객체의 액세스 권한을 설명할 수 있습니다. 예를 들어 익명 사용자에게 단일 객체에 대한 읽기 전용 권한을 부여하는 방법은 다음과 같습니다.

```
<AccessControlPolicy>
<Owner>
<ID>Owner-Cononical-CAM-User-Id</ID>
</Owner>
<AccessControlList>
<Grant>
<Grantee>
<ID>Owner-Cononical-CAM-User-Id</ID>
</Grantee>
<Permission>FULL_CONTROL</Permission>
</Grant>
<Grant>
<Grantee>
<URI>http://cam.qcloud.com/groups/global/AllUsers</URI>
</Grantee>
<Permission>READ</Permission>
</Grant>
```

```
</AccessControlList>
</AccessControlPolicy>
```

태그 기반 프로젝트 리소스 관리

최종 업데이트 날짜: : 2023-03-14 17:08:04

소개

설명 :
본 문서는 태그 시스템 하에서 어떻게 태그 및 태그 인증을 이용해 프로젝트 리소스를 관리할 수 있는지 설명합니다. 일부 기존 사용자가 이전 버전 콘솔에서 항목을 사용한 적이 있는 경우에만 적용되며, 항목 인증 방식을 통해 서브 계정에 액세스 권한을 부여합니다.

Project Management는 프로젝트를 기반으로 리소스를 중앙 집중식으로 관리하는 프로세스입니다. 프로젝트 관리를 지원하는 Tencent Cloud 제품 리소스를 프로젝트에 추가할 수 있습니다. 그 다음 Cloud Access Management(CAM) 콘솔에서 **정책 > 사용자 지정 정책 생성 > 제품 기능 또는 프로젝트 권한으로 생성**을 선택하여 프로젝트 정책을 생성할 수 있습니다. 프로젝트 정책을 프로젝트 관련 사용자 또는 사용자 그룹과 연결하여, 해당 사용자 또는 그룹이 프로젝트 리소스를 조작할 수 있는 권한을 부여할 수 있습니다.

레거시 COS(Cloud Object Storage) 콘솔에서는 프로젝트 기반의 권한 관리 작업을 지원합니다. 그러나 프로젝트 정책은 프로젝트에 포함된 모든 제품의 모든 리소스에 대한 완전한 액세스 권한을 부여합니다. COS는 **다차원 태그 및 분류 요구를 충족시키지 못하며** 또한 **세밀한 권한 관리도 처리할 수 없습니다**. 새로운 COS 콘솔에서는 프로젝트 리소스에 대한 태그 기반 권한 관리만 지원됩니다.

COS는 레거시 프로젝트 기능과의 호환성을 위해 태그 서비스를 사용합니다. 태그 서비스 시스템에서 프로젝트는 `project` 태그 키를 가진 특수한 태그입니다. 여전히 프로젝트를 만들고 프로젝트 콘솔에서 해당 프로젝트에 버킷을 만들 수 있습니다. COS는 버킷 생성 중에 해당 버킷의 프로젝트 연관성을 태그 서비스에 자동으로 두 번 기록하여 콘솔에 표시할 수 있도록 합니다.

설명 :

- 버킷을 분류 관리하려면 프로젝트 경로가 아닌 직접 태그를 통해 버킷을 관리하여 권한 제어 및 배분 등 작업을 수행하시길 권장합니다. 콘솔에 태그를 추가하는 방법은 버킷 태그 설정을 참조하십시오.
- 프로젝트에 대한 자세한 내용은 프로젝트 및 태그를 참고하십시오. 태그 서비스에 대한 자세한 내용은 태그를 참고하십시오.
- 태그의 이점에 대한 자세한 내용은 태그 사용의 장점을 참고하십시오.

프로젝트에 대한 서브 계정 액세스 권한 부여

아래 단계에 따라 프로젝트에 대한 서브 계정 액세스 권한을 부여할 수 있습니다.

1. 프로젝트 관리 콘솔에 로그인하여 프로젝트를 생성하고 이름을 지정한 후 제출합니다. 그 다음 그 아래에 버킷 또는 CVM 인스턴스와 같은 리소스를 생성합니다.
   스토리지 또는 컴퓨팅 리소스가 포함된 프로젝트가 이미 있는 경우 이 단계를 건너뛸 수 있습니다.
2. 프로젝트를 생성하고 해당 프로젝트에 리소스를 바인딩한 후 정책 관리 페이지로 이동하여 **사용자 지정 정책 생성 > 태그로 권한 부여**를 클릭하고, COS와 부여할 작업 권한을 선택한 후 해당 프로젝트 태그를 선택한 후, 서브 계정에게 해당 프로젝트의 모든 리소스에 대한 액세스 권한을 부여합니다.
3. **다음**을 클릭하고 선택한 사용자/사용자 그룹/역할에 권한을 부여한 후 **완료**를 클릭합니다.

> 설명 :
>
> - 기본 정책은 서브 계정이 프로젝트 태그 아래의 모든 리소스에 액세스할 수 있도록 권한을 부여합니다. 서브 계정이 태그 아래의 특정 리소스에서 지정된 작업만 수행하도록하려면 구문 구조에서 안내하는 대로 정책 구문의 `action` (작업 지정)과 `resource` (리소스 지정)를 변경할 수 있습니다.
> - 서브 계정이 버킷을 생성하도록 하려면, `PUT Bucket` 권한을 부여해야 합니다. 정책 관리 페이지에서 **사용자 지정 정책 생성 > 정책 생성기로 생성**을 클릭하고, 서브 계정에 해당 권한을 부여합니다.

권한 부여 방식을 선택하는 방법

최종 업데이트 날짜: : 2022-03-09 17:26:07

버킷 정책, 사용자 정책, 버킷 ACL 및 객체 ACL의 차이점

사용자 정책, 버킷 정책, 버킷 ACL 및 객체 ACL 간의 차이점은 표1에 나와 있습니다.

- 사용자 정책은 사용자 기반 권한 부여 방식이고 버킷 정책, 버킷 ACL 및 객체 ACL은 리소스 기반 권한 부여 방식입니다.
- 사용자 정책 및 버킷 정책 권한 부여는 액세스 정책 언어를 기반으로 하며 권한 제어가 보다 효율적입니다. 권한 부여는 각 작업(action)에 구체화될 수 있으며 인증 효과(effect)는 deny 또는 allow일 수 있습니다. 버킷 ACL과 객체 ACL 모두 액세스 제어 목록(ACL)을 기반으로 인증되며, 권한 제어의 효율 정도가 상대적으로 낮고 사용이 간단하지만 기본적인 읽기/쓰기 권한만 지원합니다.
- 사용자 정책은 익명 사용자에 대한 권한 부여를 지원하지 않습니다. 버킷 정책, 버킷 ACL 및 객체 ACL은 익명 사용자 권한 부여를 지원합니다.
- 사용자 정책은 CAM(Cloud Access Management) 콘솔에서 설정하고 버킷 정책, 버킷 ACL 및 객체 ACL은 COS(Cloud Object Storage) 콘솔에서 설정합니다.

표1 권한 부여 방식별 차이점 비교

<table>
<tr><th colspan="2">비교 항목</th><th>사용자 정책</th><th>버킷 정책</th><th>버킷 ACL</th><th>객체 ACL</th></tr>
<tr><td colspan="2">분류</td><td>사용자 기반 인증</td><td>리소스 기반 인증</td><td>리소스 기반 인증</td><td>리소스 기반 인증</td></tr>
<tr><td colspan="2">권한 제어 유연성</td><td>유연성 높음</td><td>유연성 높음</td><td>유연성 낮음</td><td>유연성 낮음</td></tr>
<tr><td colspan="2">콘솔 설정</td><td>CAM 콘솔</td><td>COS 콘솔</td><td>COS 콘솔</td><td>COS 콘솔</td></tr>
<tr><td rowspan="3">액세스 제어 요소</td><td rowspan="2">사용자</td><td rowspan="2">이 계정에서 관리하는 모든 CAM 신분(서브 계정, 역할 등)</td><td>서브 계정, 루트 계정, 익명 사용자</td><td rowspan="2">서브 계정, 기타 루트 계정, 익명 사용자</td><td rowspan="2">서브 계정, 기타 루트 계정, 익명 사용자</td></tr>
<tr><td>서브 계정 Tencent Cloud 서비스, 역할 등</td></tr>
<tr><td>교차 계정 권한 부여는 먼저 협업 파트너로 추가해야 함</td><td colspan="3">교차 계정 권한 부여 직접 지원</td></tr>
</table>

	효과	허용+거부	허용+거부	허용	허용
	리소스	모든 클라우드 리소스, 모든 COS 버킷, 버킷 지정(접두사, 객체 등)	버킷 지정(접두사, 객체 등)	전체 버킷	객체 지정
	동작	각 구체적인 동작	각 구체적인 동작 (버킷 생성, 버킷 나열 제외)	단순화된 읽기/쓰기 권한	단순화된 읽기/쓰기 권한
	조건	지원	지원	미지원	미지원

적절한 권한 부여 방식을 선택하는 방법은 무엇입니까?

표1에 나열된 차이점을 통해 다양한 인증 방법의 장단점을 명확하게 볼 수 있으며 필요에 따라 적절한 권한 부여 방법을 선택할 수 있습니다.
그러나 어떤 방법을 선택하든 가능한 한 통일되게 유지하는 것이 좋습니다. 버킷 정책, 사용자 정책 및 ACL이 증가함에 따라 권한 관리 난이도가 높아집니다.

어떤 경우 ACL을 선택해야 합니까?

> 주의 :
> 공개 익명 사용자 액세스(공개 읽기)는 매우 위험한 작업으로 트래픽 도용의 리스크가 있으므로 공개 읽기를 사용해야 하는 경우 보안을 위해 링크 도용 방지 설정을 할 수 있습니다.

간단한 액세스 권한을 설정하거나 버킷 및 객체에 대한 익명 액세스를 개방해야 하는 경우 ACL을 선택하십시오. 그러나 더 많은 경우에는 효율성이 높은 버킷 정책 또는 사용자 정책을 먼저 사용하는 것을 추천합니다.

- 단순 액세스 권한만 설정합니다.
- 콘솔에서 액세스 권한을 빠르게 설정합니다.
- 모든 익명의 인터넷 사용자에게 객체, 디렉터리 또는 버킷을 개방하며 ACL 작업이 더 편리합니다.
- 계정당 ACL은 1000개까지만 설정할 수 있으며, 최댓값을 초과할 경우 버킷 정책이나 사용자 정책을 대신 선택하십시오.

어떤 경우 사용자 정책을 선택해야 합니까?

사용자가 무엇을 할 수 있는지에 중점을 두는 경우 사용자 정책을 사용하는 것이 좋습니다. CAM 사용자와 그들이 속한 사용자 그룹의 권한을 확인하여 CAM 사용자가 무엇을 할 수 있는지 알아볼 수 있습니다. 권장 시나리오는 다음과

같습니다.

- 버킷 생성 및 버킷 나열과 같은 COS 서비스 레벨 권한을 설정하려는 경우.
- 루트 계정의 모든 COS 버킷 및 객체를 사용해야 하는 경우.
- 루트 계정으로 다수의 CAM 사용자에게 동일한 권한을 부여하려는 경우.

어떤 경우 버킷 정책을 선택해야 합니까?

COS 버킷에 누가 액세스할 수 있는지에 중점을 두는 경우 버킷 정책을 사용하는 것이 좋습니다. 버킷을 조회하고 버킷 정책을 확인하여 액세스할 수 있는 사용자를 확인할 수 있습니다. 권장 시나리오는 다음과 같습니다.

- 버킷에 대한 개별 권한 부여를 진행하는 경우.
- ACL 대비, 개별 작업(action)에 대한 구체적인 권한 부여가 가능합니다. 권한 부여 효과(effect)는 deny 또는 allow여야 합니다.
- 사용자 정책 대비, 버킷 정책은 교차 계정 권한 부여 및 익명 사용자 권한 부여를 지원합니다.

액세스 정책 언어
액세스 정책 언어 개요

최종 업데이트 날짜: : 2023-01-13 14:24:23

주의 :
서브 계정 또는 협업 파트너에게 액세스 정책을 추가할 때 비즈니스 요구 사항을 충족하는 데 필요한 최소한의 API 액세스 권한을 부여해야 합니다. 모든 리소스 `(resource:*)` 또는 모든 작업 `(action:*)` 에 과도한 액세스 권한을 부여하는 것과 관련된 데이터 보안 리스크가 있을 수 있습니다.

개요

JSON 기반 액세스 정책 언어를 사용하는 액세스 정책은 COS(Cloud Object Storage) 리소스에 대한 액세스 권한을 부여하는 데 사용됩니다. 액세스 정책 언어를 통해 지정된 COS 리소스에 대한 조치를 수행하도록 지정된 위탁자(principal)에게 권한을 부여할 수 있습니다.

액세스 정책 언어는 버킷 정책(Bucket Policy)의 기본 요소와 용법을 설명합니다. 정책 언어에 관한 자세한 설명은 Concepts를 참조하십시오.

액세스 정책 요소

액세스 정책 언어는 다음의 기본 요소를 포함합니다.

- **위탁자(principal)**: 정책에 의해 권한이 부여되는 엔티티입니다. 예시로 사용자(루트 계정, 서브 계정, 익명 사용자), 사용자 그룹 등이 있습니다. 이 요소는 사용자 액세스 정책이 아닌 버킷 액세스 정책에 유효합니다.
- **명령(statement)**: 하나 또는 여러 개의 권한의 자세한 정보를 기술하는 데 사용합니다. 이 요소에는 효력, 작업, 리소스, 조건 등의 기타 여러 요소의 권한 또는 권한 집합이 포함됩니다. 하나의 정책에는 하나의 명령 요소만 가지고 있습니다.
- **효력(effect)**: 선언으로 발생한 결과가 “허용”인지 “명시적 거부”인지 기술합니다. allow 및 deny 두 가지 상황을 포함합니다. 이 요소는 필수 항목입니다.
- **작업(action)**: 허용 또는 거부의 작업을 기술하는 데 사용합니다. 작업은 API(접두사 name으로 표시) 또는 기능 집합(특정한 API 구성, 접두사 permid로 표시)이 가능합니다. 이 요소는 필수 항목입니다.
- **리소스(resource)**: 권한 부여의 구체적인 데이터를 기술합니다. 리소스는 6단식 기술을 사용하며, 모든 제품마다 리소스의 정보 정의의 차이가 있습니다. 리소스 정보를 어떻게 지정할 지에 대한 것은 작성한 리소스 선언에 대한 제품 문서를 참조하십시오. 이 요소는 필수 항목입니다.

- **조건(condition)**: 정책의 효력이 발생하는 규제 조건을 기술합니다. 조건에는 오퍼레이터, 작업 키와 작업 값 구성이 포함됩니다. 조건 값은 시간, IP 주소 등의 정보를 포함합니다. 어떤 서비스는 조건에 기타 값을 지정하는 것을 허용합니다. 이 요소는 필수 항목입니다.

요소 사용 방법

지정 위탁자(principal)

principal은 사용자, 계정, 서비스 또는 리소스 액세스가 허용되거나 허용되지 않는 다른 엔터티를 지정하는 데 사용됩니다. principal은 버킷에서만 작동하며 이러한 정책은 특정 사용자에게 직접 추가되기 때문에 사용자 정책에는 필요하지 않습니다. 다음 예시에서는 principal을 지정합니다.

```
"principal":{
"qcs":[
"qcs::cam::uin/100000000001:uin/100000000001"
]
}
```

익명 사용자에 권한 부여

```
"principal":{
"qcs":[
"qcs::cam::anonymous:anonymous"
]
}
```

루트 계정 UIN 100000000001에 권한 부여

```
"principal":{
"qcs":[
"qcs::cam::uin/100000000001:uin/100000000001"
]
}
```

서브 계정 UIN 100000000011(루트 계정 UIN은 100000000001)에 권한 부여

> 주의 :
> 작업을 수행하기 전 서브 계정이 루트 계정의 서브 계정 리스트에 추가되었는지 확인하십시오.

```
"principal":{
"qcs":[
"qcs::cam::uin/100000000001:uin/100000000011"
]
}
```

지정 효력

리소스에 대한 액세스 권한을 명시적으로 부여(허용)하지 않는 경우 액세스가 암묵적으로 거부됩니다. 또한 리소스에 대한 액세스를 명시적으로 거부(deny)할 수 있어 이런 방식으로 리소스에 대한 액세스를 거부할 수도 있습니다. 이는 다른 정책에서 액세스를 허용하더라도 사용자가 리소스에 액세스할 수 없도록 하기 위해 수행할 수 있습니다. 다음 예시는 허용 효과를 지정합니다.

```
"effect" : "allow"
```

지정 작업

COS는 정책에서 특정한 COS 작업을 지정할 수 있으며, 지정 작업과 API 요청 발송 작업은 완전히 일치합니다. 다음은 일부 버킷 작업과 객체 작업입니다. 자세한 내용은 API 작업 리스트 문서를 참조하십시오.

버킷 작업

설명	해당 API 인터페이스
name/cos:GetService	GET Service
name/cos:GetBucket	GET Bucket (List Objects)
name/cos:PutBucket	PUT Bucket
name/cos:DeleteBucket	DELETE Bucket

객체 작업

설명	해당 API 인터페이스
name/cos:GetObject	GET Object
name/cos:PutObject	PUT Object
name/cos:HeadObject	HEAD Object
name/cos:DeleteObject	DELETE Object

허용된 작업의 예시는 다음과 같습니다.

```
"action":[
"name/cos:GetObject",
"name/cos:HeadObject"
]
```

지정 리소스

리소스(resource)요소는 하나 또는 여러 작업 객체를 기술합니다. 예시로 COS 버킷 또는 객체 등이 있으며, 모든 리소스는 6단식 기술을 사용합니다.

```
qcs:project_id:service_type:region:account:resource
```

매개변수 설명은 다음과 같습니다.

매개변수	설명	필수 여부
qcs	는 qcloud service의 약칭으로 Tencent Cloud의 클라우드 서비스를 의미합니다.	필수
project_id	항목 정보이며 CAM 초기 로직에만 호환됩니다.	선택 가능
service_type	COS와 같은 제품 약칭입니다.	필수
region	리전 정보입니다. Tencent Cloud COS가 지원하는 리전 및 액세스 도메인을 참조하십시오.	필수
account	리소스 소유자의 루트 계정 정보이며, 리소스 소유자를 설명하는 두가지 방식을 지원합니다. 첫 번째 방식은 uin 방식, 즉 루트 계정의 UIN 계정이며 `uin/${OwnerUin}` 로 표시됩니다(예: uin/100000000001). 또 다른 방식은 uid 방식, 즉 루트 계정의 APPID이며 `uid/${appid}` 로 표시됩니다(예: uid/1250000000). 현재 COS의 리소스 소유자는 일괄적으로 uid 방식을 사용해 표시합니다. 즉 루트 계정의 개발사 APPID입니다.	필수
resource	구체적인 리소스 세부 내용을 의미합니다. COS 서비스에서는 버킷 XML API 호출 도메인을 이용해 설명합니다.	필수

다음은 지정 버킷 examplebucket-1250000000의 예시입니다.

```
"resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"]
```

다음은 지정 버킷 examplebucket-1250000000의 /folder/ 폴더에 있는 모든 객체 예시입니다.

```
"resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/folde
r/*"]
```

다음은 지정 버킷 examplebucket-1250000000에 있는 /folder/exampleobject 객체 예시입니다.

```
"resource": ["qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/folde
r/exampleobject"]
```

지정 조건

권한을 부여 받을 경우 액세스 정책 언어로 조건을 지정할 수 있습니다. 사용자 액세스 제한, 권한 부여 시간 제한 등이 그 예입니다. 다음은 현재 지원하는 조건 오퍼레이터 리스트 및 통용되는 조건 키, 예시 등의 정보입니다.

조건 오퍼레이터	의미	조건 이름	예시
ip_equal	IP 일치	qcs:ip	{"ip_equal":{"qcs:ip ":"10.121.2.0/24"}}
ip_not_equal	IP 불일치	qcs:ip	{"ip_not_equal":{"qcs:ip ":["10.121.1.0/24", "10.121.2.0/24"]}}

다음은 액세스 IP가 IP 대역 내에 있는 10.121.2.0/24 예시입니다.

```
"ip_equal":{"qcs:ip ":"10.121.2.0/24"}
```

다음은 액세스 IP가 101.226.100.185와 101.226.100.186인 예시입니다.

```
"ip_equal":{
"qcs: ip": [
"101.226.100.185",
"101.226.100.186"
]
}
```

실제 사례

루트 계정이 익명 사용자를 허용하고 액세스 출처 IP가 101.226.100.185/101.226.100.186인 경우 화남 리전의 버킷이 examplebucket-1250000000인 객체에 대해 GET(다운로드)과 HEAD 작업을 실행하여 인증할 필요가 없습니다. 자세한 내용은 권한 설정 관련 사례를 참고하십시오.

```
{
"version": "2.0",
"principal":{
```

```
"qcs":[
"qcs: : cam: : anonymous: anonymous"
]
},
"statement": [
{
"action":[
"name/cos: GetObject",
"name/cos: HeadObject"
],
"condition":{
"ip_equal":{
"qcs: ip": [
"101.226.100.185",
"101.226.100.186"
]
}
},
"effect": "allow",
"resource":[
"qcs: : cos: ap-guangzhou: uid/1250000000: examplebucket-1250000000.ap-guangzhou.
myqcloud.com/*"
]
}
]
}
```

조건

최종 업데이트 날짜: : 2023-01-13 14:24:23

소개

조건은 액세스 정책 언어의 일부입니다. 완전한 조건에는 다음 요소가 포함됩니다.

- 조건 키: 조건의 유형을 지정합니다(예: 사용자 액세스 소스 IP 및 권한 부여 시간).
- 조건 연산자: 조건 결정 방법을 지정합니다.
- 조건 값: 조건 키의 값을 지정합니다.

자세한 내용은 Conditions를 참고하십시오.

> 설명 :
>
> - 버킷 정책을 작성 시 조건 키를 사용할 때 최소 권한 원칙을 준수하고, 해당 조건 키를 해당 요청(action)에만 추가하며, 작업 지정(action) 시 "*" 와일드 카드를 사용하지 마십시오. 와일드카드를 사용하면 요청이 실패합니다.
> - 액세스 관리(CAM) 콘솔을 사용하여 정책을 생성할 때, version, principal, statement, effect, action, resource, condition 구문 요소의 첫 번째 글자는 대문자이거나 모두 소문자여야 합니다.

조건 예시

다음 버킷 정책 예시에서 조건(condition)은 `cos:PutObject` 권한 부여 작업이 10.217.182.3/24 또는 111.21.33.72/24 IP 범위에서만 완료될 수 있음을 지정합니다.

- **조건 키**는 `qcs:ip` 이며 조건 유형이 IP임을 나타냅니다.
- **조건 연산자**는 `ip_equal` 로, 조건 판단 방법이 IP 주소 일치 여부를 판단하는 것임을 나타냅니다.
- **조건 값**은 조건 결정을 위해 지정된 값을 나열하는 `["10.217.182.3/24","111.21.33.72/24"]` 배열입니다. 사용자의 IP가 어레이의 지정된 IP 범위에 있는 경우 조건은 true로 결정됩니다.

```
{
"version":"2.0",
"statement":[
{
"principal":{
```

```
"qcs":[
"qcs::cam::uin/1250000000:uin/1250000001"
]
},
"effect":"allow",
"action":[
"name/cos:PutObject"
],
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
],
"condition":{
"ip_equal":{
"qcs:ip":[
"10.217.182.3/24",
"111.21.33.72/24"
]
}
}
}
]
}
```

COS에서 지원하는 조건 키

> 설명 :
> 현재 `tls-version` 조건 키는 베이징 리전에서만 지원됩니다. 추후 다른 리전에서도 지원될 예정입니다.

COS는 IP, VPC 및 HTTPS를 포함한 모든 요청에 적용 가능한 조건 키와 요청 헤더 및 요청 매개변수의 조건 키, 일반적으로 요청 헤더 또는 요청 매개변수를 전달해야 하는 요청에만 적용할 수 있는 두 가지 유형의 조건 키를 지원합니다. 이러한 조건 키에 대한 설명 및 사용 사례는 Descriptions and Use Cases of Condition Keys를 참고하십시오.

> 설명 :
> 조건 및 조건 키는 사용자 요청의 액세스 관리에만 적용됩니다. 수명 주기 및 버킷 복제 규칙이 유효한 경우 삭제 및 복제와 같은 작업은 사용자가 아닌 COS에서 시작하므로 조건 키의 해당 범위에 포함되지 않습니다.

모든 요청에 적용 가능한 조건 키

모든 요청에 적용할 수 있는 조건 키는 각각 `qcs:ip`, `qcs:vpc`, `cos:secure-transport`로 요청의 소스 IP 범위, 요청의 소스 VPC ID, HTTPS 사용 여부를 나타냅니다.

조건 키	적용 요청	의미	유형
cos:secure-transport	모든 요청	요청의 HTTPS 사용 여부	Boolean
qcs:ip	모든 요청	요청의 소스 IP 범위	IP
qcs:vpc	모든 요청	요청의 소스 VPC ID	String
cos:tls-version	모든 https 요청	https 요청에 사용되는 TLS 버전	Numeric

요청 헤더 및 요청 매개변수의 조건 키

요청마다 요청 헤더(Header) 및 요청 매개변수(Param)가 다르기 때문에 요청 헤더 및 요청 매개변수의 조건 키는 이러한 요청 헤더 또는 요청 매개변수를 포함하는 요청에만 적용할 수 있습니다.

예를 들어 조건 키 `cos:content-type`은 요청 헤더 `Content-Type`을 사용해야 하는 업로드 요청(예: PutObject)에 적용할 수 있는 반면 조건 키 `cos:response-content-type`은 GetObject 요청에만 적용할 수 있습니다. GetObject 요청만 요청 매개변수 `response-content-type`을 지원합니다.

아래 표에는 요청 헤더 및 요청 매개변수의 조건 키와 해당하는 해당 요청이 나열되어 있습니다.

조건 키	적용 요청	요청 헤더 또는 요청 매개변수 확인	유형
cos:x-cos-storage-class	PutObject PostObject InitiateMultipartUpload AppendObject	요청 헤더: x-cos-storage-class	String
cos:versionid	GetObject DeleteObject PostObjectRestore PutObjectTagging GetObjectTagging DeleteObjectTagging HeadObject	요청 매개변수: versionid	String
cos:prefix	GetBucket（List Objects） GET Bucket Object versions List Multipart Uploads ListLiveChannels	요청 매개변수: prefix	String

조건 키	적용 요청	요청 헤더 또는 요청 매개변수 확인	유형
cos:x-cos-acl	PutObject PostObject PutObjectACL PutBucket PutBucketACL AppendObject Initiate Multipart Upload	요청 헤더: x-cos-acl	String
cos:content-length	이 요청 헤더는 적용 가능한 범위가 넓으며 일반적으로 요청 본문이 있는 요청	요청 헤더: Content-Length	Numeric
cos:content-type	이 요청 헤더는 적용 가능한 범위가 넓으며 일반적으로 요청 본문이 있는 요청	요청 헤더: Content-Type	String
cos:response-content-type	GetObject	요청 매개변수: response-content-type	String
qcs:request_tag	PutBucket PutBucketTagging	요청 헤더: x-cos-tagging 요청 매개변수: tagging	String

조건 연산자

COS는 문자열(String), 숫자(Numeric), 부울(Boolean) 및 IP 유형의 조건 키에 적용할 수 있는 다음 조건 연산자를 지원합니다.

조건 연산자	설명	유형
string_equal	같은 문자열(대소문자 구분)	String
string_not_equal	같지 않은 문자열(대소문자 구분)	String
string_like	유사한 문자열(대소문자 구분), 현재 와일드카드(`*`)를 문자열에 접두사 또는 접미사로 사용 가능, 예시 `image/*`	String
ip_equal	IP 같음	IP
ip_not_equal	IP가 같지 않음	IP
numeric_equal	같은 숫자	Numeric

조건 연산자	설명	유형
numeric_not_equal	같지 않은 숫자	Numeric
numeric_greater_than	큰 숫자	Numeric
numeric_greater_than_equal	크거나 같은 숫자	Numeric
numeric_less_than	작은 숫자	Numeric
numeric_less_than_equal	작거나 같은 숫자	Numeric

_if_exist의 의미

이전 조건 연산자의 끝에 `_if_exist` 를 추가하여 `string_equal_if_exist` 와 같은 새 조건 연산자를 구성할 수 있습니다. `_if_exist` 가 있는 조건 연산자와 없는 조건 연산자의 차이점은 다음과 같습니다.

- `string_equal` 과 같이 `_if_exist` 가 없는 조건 연산자의 경우 요청에 지정된 요청 헤더 또는 매개변수가 포함되어 있지 않으면 기본적으로 조건이 충족된(`False`) 것으로 간주됩니다.
- `string_equal_if_exist` 와 같이 `_if_exist` 가 있는 조건 연산자의 경우 요청에 지정된 요청 헤더 또는 매개변수가 포함되어 있지 않으면 기본적으로 조건이 충족된(`True`) 것으로 간주됩니다.

License 요청 예시

예시1: 지정된 버전의 객체 다운로드 허용

이 예시의 버킷 정책에서 Effect는 allow이며 요청 매개변수 versionid가 'MTg0NDUxNTc1NjIzMTQ1MDAwODg'인 GetObject 요청을 허용합니다. 권한 부여 allow 정책에 따라 조건이 충족되면(True) 요청이 허용됩니다. 조건이 충족되지 않으면(False) 요청이 허용되지 않고 실패합니다.

```
{
"version":"2.0",
"statement":[
{
"principal":{
"qcs":[
"qcs::cam::uin/1250000000:uin/1250000001"
]
},
"effect":"allow",
"action":[
"name/cos:GetObject"
],
```

```
"condition":{
"string_equal":{
"cos:versionid":"MTg0NDUxNTc1NjIzMTQ1MDAwODg"
}
},
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
]
}
]
}
```

아래 표는 조건 연산자 `string_equal` 및 `string_equal_if_exist` 의 condition 충족 및 요청 허용 세부 정보를 나열합니다.

조건 연산자	요청	condition 충족	요청 허용 여부
string_equal	versionid 가 없는 경우	FALSE	No
string_equal_if_exist	versionid 가 없는 경우	TRUE	Yes
string_equal	값이 지정된 versionid 사용	TRUE	Yes
string_equal_if_exist	값이 지정된 versionid 사용	TRUE	Yes
string_equal	값이 지정되지 않은 versionid 사용	FALSE	No
string_equal_if_exist	값이 지정되지 않은 versionid 사용	FALSE	No

예시2: 지정된 버전의 객체 다운로드 금지

이 예시의 버킷 정책에서 Effect는 deny이며 요청 매개변수 versionid가 'MTg0NDUxNTc1NjIzMTQ1MDAwODg'인 GetObject 요청을 허용하지 않습니다. deny 권한 부여 정책에 따라 조건이 충족되면(True) 요청이 실패합니다. 조건이 충족되지 않으면(False) 요청이 거부되지 않습니다.

```
{
"version":"2.0",
"statement":[
{
"principal":{
"qcs":[
"qcs::cam::uin/1250000000:uin/1250000001"
]
},
"effect":"deny",
"action":[
```

```
"name/cos:GetObject"
],
"condition":{
"string_equal":{
"cos:versionid":"MTg0NDUxNTc1NjIzMTQ1MDAwODg"
}
},
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
]
}
]
}
```

아래 표는 조건 연산자 `string_equal` 및 `string_equal_if_exist` 의 condition 충족 및 요청 거부 세부 정보를 나열합니다.

조건 연산자	요청	condition 충족	요청 거부 여부
string_equal	versionid 가 없는 경우	FALSE	No
string_equal_if_exist	versionid 가 없는 경우	TRUE	Yes
string_equal	값이 지정된 versionid 사용	TRUE	Yes
string_equal_if_exist	값이 지정된 versionid 사용	TRUE	Yes
string_equal	값이 지정되지 않은 versionid 사용	FALSE	No
string_equal_if_exist	값이 지정되지 않은 versionid 사용	FALSE	No

관련 설명

특수 문자에는 urlencode 필요

요청 매개변수의 특수 문자에는 모두 urlencode가 필요하므로 요청 매개변수의 조건 키를 사용하는 버킷 정책에도 urlencode가 필요합니다. 예를 들어 버킷 정책에서 `cos:response-content-type` 조건 키를 사용하려는 경우 버킷 정책에 입력하기 전에 조건 값 "image/jpeg"를 "image%2Fjpeg"로 urlencode해야 합니다.

최소 권한 원칙 및 * 와일드카드 없음

조건 키를 사용할 때 최소 권한 원칙을 준수하고 권한을 설정하려는 action만 추가하고 '*' 와일드카드를 사용하지 마십시오. '*' 와일드카드를 잘못 사용하면 일부 요청이 실패할 수 있습니다. 아래 예에서 GetObject 이외의 요청은 요청 매개변수 response-content-type 사용을 지원하지 않습니다.

deny + string_equal_if_exist의 경우 조건 키가 요청에 없으면 기본적으로 true로 간주됩니다. 따라서 PutObject 및 PutBucket과 같은 요청을 시작하면 deny statement가 충족되고 요청이 거부됩니다.

allow + string_equal의 경우 조건 키가 요청에 없으면 기본적으로 false로 간주됩니다. 따라서 PutObject 및 PutBucket과 같은 요청을 시작할 때 allow statement가 충족되지 않고 요청이 허용되지 않습니다.

```
{
"version":"2.0",
"statement":[
{
"principal":{
"qcs":[
"qcs::cam::uin/1250000000:uin/1250000001"
]
},
"effect":"allow",
"action":[
"*"
],
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
],
"condition":{
"string_equal":{
"cos:response-content-type":"image%2Fjpeg"
}
}
},
{
"principal":{
"qcs":[
"qcs::cam::uin/1250000000:uin/1250000001"
]
},
"effect":"deny",
"action":[
"*"
],
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
],
"condition":{
"string_not_equal_if_exist":{
"cos:response-content-type":"image%2Fjpeg"
}
}
```

```
}
]
}
```

또는 allow+string_equal_if_exist 및 deny + string_not_equal을 사용하여 response-content-type 요청 매개변수 없이 요청을 허용할 수 있습니다.

deny + string_equal의 경우 조건 키가 요청에 없으면 기본적으로 false로 간주됩니다. 따라서 PutObject 및 PutBucket과 같은 요청을 시작할 때 deny statement가 충족되지 않고 요청이 거부되지 않습니다.

allow + string_equal_if_exist의 경우 조건 키가 요청에 없으면 기본적으로 true로 간주됩니다. 따라서 PutObject 및 PutBucket과 같은 요청을 시작할 때 allow statement가 충족되고 요청이 허용됩니다.

그러나 이러한 방식으로 조건 연산자를 사용하면 GetObject 요청의 response-content-type 요청 매개변수 전달 여부를 제한할 수 없습니다. response-content-type 요청 매개변수가 없는 GetObject 요청은 다른 요청과 마찬가지로 기본적으로 허용됩니다. GetObject 요청이 response-content-type 요청 매개변수를 전달하는 경우에만 지정된 조건을 사용하여 요청 매개변수의 내용이 조건부 승인을 구현할 것으로 예상하는 내용과 동일한지 확인할 수 있습니다.

```
{
"version":"2.0",
"statement":[
{
"principal":{
"qcs":[
"qcs::cam::uin/1250000000:uin/1250000001"
]
},
"effect":"allow",
"action":[
"*"
],
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
],
"condition":{
"string_equal_if_exist":{
"cos:response-content-type":"image%2Fjpeg"
}
}
},
{
"principal":{
"qcs":[
"qcs::cam::uin/1250000000:uin/1250000001"
]
},
```

```
"effect":"deny",
"action":[
"*"
],
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
],
"condition":{
"string_not_equal":{
"cos:response-content-type":"image%2Fjpeg"
}
}
}
]
}
```

따라서 보다 안전한 방법은 최소 권한 원칙을 준수하고 '*' 와일드카드를 사용하지 않고 GetObject 요청에 대한 action 을 제한하는 것입니다.

아래 예시 정책에서 볼 수 있듯이 정책 조건은 값이 "image%2Fjpeg"인 response-content-type 요청 매개변수를 전달하는 GetObject 요청에 대해서만 권한 부여가 수행되도록 엄격하게 지정합니다.

다른 요청은 이 예시에서 정책의 영향을 받지 않으며 최소 권한 원칙에 따라 별도로 권한을 부여할 수 있습니다.

```
{
"version":"2.0",
"statement":[
{
"principal":{
"qcs":[
"qcs::cam::uin/1250000000:uin/1250000001"
]
},
"effect":"allow",
"action":[
"name/cos:GetObject"
],
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
],
"condition":{
"string_equal":{
"cos:response-content-type":"image%2Fjpeg"
}
}
},
{
```

```
"principal":{
"qcs":[
"qcs::cam::uin/1250000000:uin/1250000001"
]
},
"effect":"deny",
"action":[
"name/cos:GetObject"
],
"resource":[
"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*"
],
"condition":{
"string_not_equal_if_exist":{
"cos:response-content-type":"image%2Fjpeg"
}
}
}
]
}
```

요청 방법 소개
영구 키를 사용한 COS 액세스

최종 업데이트 날짜: : 2022-03-09 17:29:34

배경 소개

RESTful API를 통해 COS(Cloud Object Storage)에 대한 HTTP 익명 요청 또는 HTTP 서명 요청을 제안할 수 있습니다. 익명 요청은 일반적으로 정적 웹 사이트 호스팅과 같이 공개 액세스가 필요한 시나리오에서 사용되며 대부분의 시나리오는 서명된 요청을 통해 완료해야 합니다.

서명된 요청은 익명 요청에 비해 서명 값을 하나 더 가지고 있으며 서명은 키(SecretId/SecretKey)와 요청 정보를 기반으로 암호화되어 생성된 문자열입니다. SDK는 자동으로 서명을 계산하므로 사용자 정보를 초기화할 때 키를 설정하기만 하면 되며 서명 계산은 신경 쓰지 않아도 됩니다. RESTful API를 통해 발송된 요청의 경우 서명 알고리즘에 따라 서명을 계산하고 요청에 추가해야 합니다.

영구 키 가져오기

CAM 콘솔의 API 키 관리 페이지에서 영구 키를 얻을 수 있으며 영구 키는 계정의 영구 신분을 나타내는 SecretId 및 SecretKey를 포함하며 만료되지 않습니다.

- SecretId: API 호출자를 식별하기 위해 사용합니다.
- SecretKey: 서명 문자열을 암호화하고 서버에서 서명 문자열을 인증하는 데 사용되는 키입니다.

영구 키를 사용한 COS 액세스

API 요청을 통해 COS 액세스

API 요청을 사용할 때는 프라이빗 버킷에 대해 서명된 요청을 사용해야 합니다. 영구 키를 통해 서명을 생성하고 이를 Authorization 헤더에 넣어 서명 요청을 형성합니다. 요청이 COS로 전송되고 COS는 서명이 요청과 일치하는지 인증합니다.

> 설명 :
> 서명 생성 알고리즘이 복잡하므로 SDK를 직접 사용하여 요청을 제안하고 이 링크를 생략하는 것이 좋습니다.

1. 영구 키로 서명 생성
   서명 알고리즘에 대한 소개는 서명 요청 문서를 참고하십시오. COS에서도 서명 생성 툴을 제공하고 있으며, SDK를 통해서도 서명을 생성할 수 있습니다. SDK 서명 구현을 참고하십시오. 자체 프로그램을 작성하여 서명을 생성할 수도 있지만 서명 알고리즘은 더 복잡하기 때문에 이 방법은 일반적으로 권장되지 않습니다.
2. Authorization 헤더 입력
   API 요청을 제안할 때 표준 Http Authorization 헤더에 서명을 입력합니다. 다음은 GetObject 요청의 예시입니다.

```
GET /<ObjectKey> HTTP/1.1
Host: <BucketName-APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
Authorization: q-sign-algorithm=sha1&q-ak=SecretId&q-sign-time=KeyTime&q-key-time=KeyTime&q-header-list=HeaderList&q-url-param-list=UrlParamList&q-signature=Signature
```

SDK 툴을 통해 COS 액세스

1. 영구 키로 신분 정보 초기화
   SDK 툴을 설치한 후, 먼저 사용자 신분 정보를 초기화하고 루트 계정 또는 서브 계정의 영구 키(SecretId 및 SecretKey)를 입력해야 합니다.
2. SDK를 직접 사용하여 COS 요청
   SDK 툴이 사용자를 대신하여 키를 통해 서명을 생성하고 COS에 요청을 제안하기 때문에 초기화 후 SDK 툴을 사용하여 API 요청처럼 직접 서명을 생성하는 대신 업로드 및 다운로드와 같은 기본 작업을 직접 수행할 수 있습니다.

예를 들어 다음 Java SDK 코드, 더 많은 언어 demo는 SDK 개요의 시작하기 문서를 참고하십시오.

```
// 1 사용자의 개인 정보(secretId, secretKey)를 초기화합니다.
// SECRETID와 SECRETKEY는 CAM 콘솔 https://console.cloud.tencent.com/cam/capi에 로그인하여 조회 및 관리
String secretId = "SECRETID";
String secretKey = "SECRETKEY";
COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);
// 2 bucket의 리전 설정. COS 리전의 약칭은 https://cloud.tencent.com/document/product/436/6224 참고 바랍니다.
// clientConfig에 region, https(기본값: http), 타임아웃, 프록시 등을 설정하는 set 메소드가 포함되어 있습니다. 사용 시 소스 코드 또는 FAQ의 Java SDK 부분을 참고하십시오.
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
// https 프로토콜 설정 및 사용 권장
// 버전 5.6.54부터 https가 기본적으로 사용됨
clientConfig.setHttpProtocol(HttpProtocol.https);
// 3 cos 클라이언트 생성.
COSClient cosClient = new COSClient(cred, clientConfig);
```

임시 키를 사용한 COS 액세스

최종 업데이트 날짜: : 2022-05-16 18:23:36

임시 키 소개

임시 키는 STS(Security Token Service)에서 제공하는 임시 액세스 자격 증명입니다. 임시 키는 TmpSecretId, TmpSecretKey 및 Token의 세 부분으로 구성되며 영구 키와 비교하여 임시 키는 다음과 같은 특징이 있습니다.

- 짧은 유효 시간(30min - 36h), 영구 키를 노출할 필요가 없으므로 계정 유출 리스크가 줄어듭니다.
- 임시 키를 얻을 때 policy 매개변수에 임시 권한을 설정하여 사용자의 권한 범위를 더욱 제한할 수 있습니다.

따라서 임시 키는 프런트 엔드 다이렉트 업로드 등 임시 인증 시나리오에 적합하며 영구 키에 비해 신뢰할 수 없는 사용자에게 임시 키를 배포하는 것이 더 안전합니다.

임시 키 획득

임시 키는 당사에서 제공하는 COS STS SDK 또는 STS 클라우드 API를 통해 직접 얻을 수 있습니다.
자세한 내용은 임시 키 생성 및 사용 가이드를 참고하십시오.

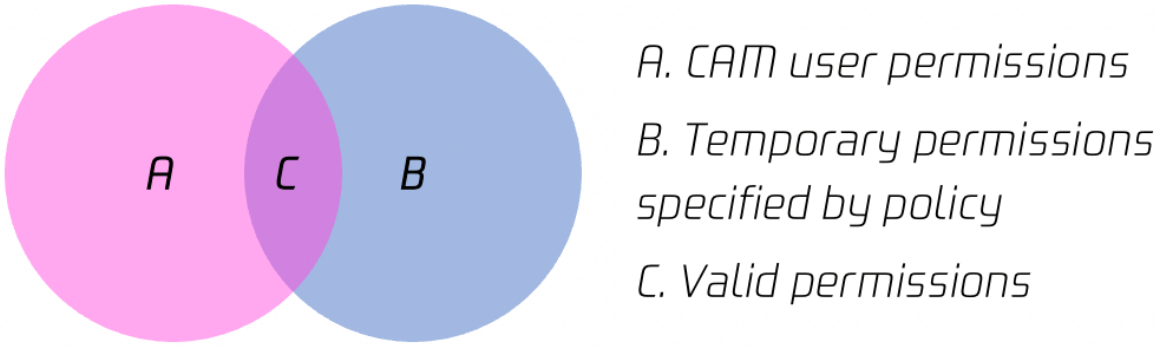
임시 키의 권한

임시 키를 신청하기 전에 CAM(Cloud Access Management) 사용자(Tencent Cloud 루트 계정 또는 서브 계정)가 있어야 하며, Policy 매개변수를 설정하여 임시 키에 임시 정책을 추가하여 사용자의 권한을 제한할 수 있습니다..

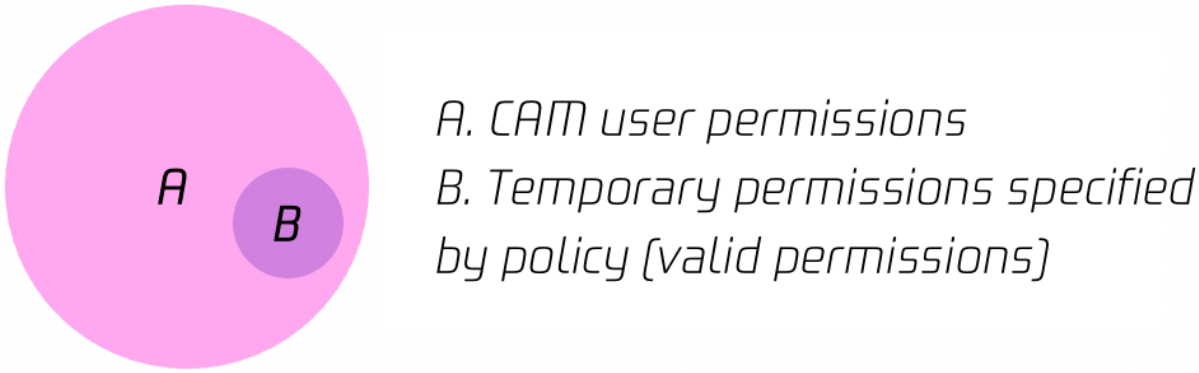
- policy 매개변수가 설정되지 않은 경우 획득한 임시 키는 CAM 사용자와 동일한 권한을 갖습니다.
- policy 매개변수가 설정되면 획득한 임시 키는 CAM 사용자 권한에서 더 나아가 policy에서 설정 범위 내에서 권한을 추가로 제한합니다.

가령 'A'는 CAM 사용자의 원래 권한을 나타내고 'B'는 policy 매개변수를 통해 임시 키에 대해 설정된 권한을 나타내며 'A'와 'B'가 교차하는 것은 임시 키의 최종 유효 권한을 나타냅니다.

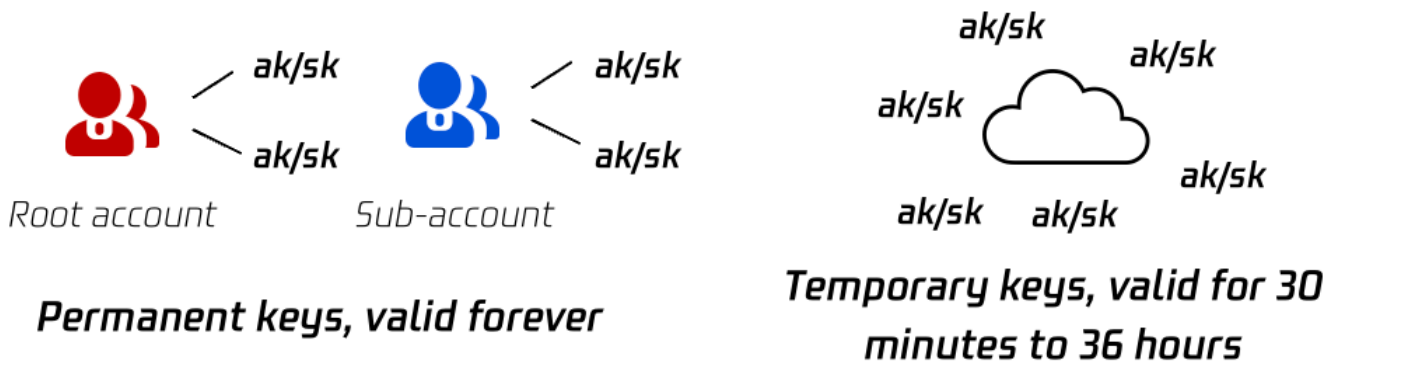
아래 이미지와 같이 CAM 사용자 권한과 policy 임시 권한의 교집합이 유효 권한입니다.



아래 이미지와 같이 policy는 CAM 사용자 권한 내에 있으며 policy는 유효 권한입니다.



임시 키를 사용한 COS 액세스



임시 키는 SecretId, SecretKey 및 Token이 있으며, 각 루트 계정과 서브 계정은 여러 개의 임시 키를 생성할 수 있습니다. 영구 키에 비해 임시 키는 30분 - 36시간 동안만 유효합니다. 임시 키는 프런트 엔드 다이렉트 업로드와 같은 임시 인증 시나리오에 적합하며 영구 키에 비해 신뢰할 수 없는 사용자에게 임시 키를 배포하는 것이 더 안전합니다. 자세한 내용은 임시 키 생성 및 사용 가이드 및 프런트 엔드 다이렉트 업로드를 위한 임시 키 사용 가이드를 참고하십시오.

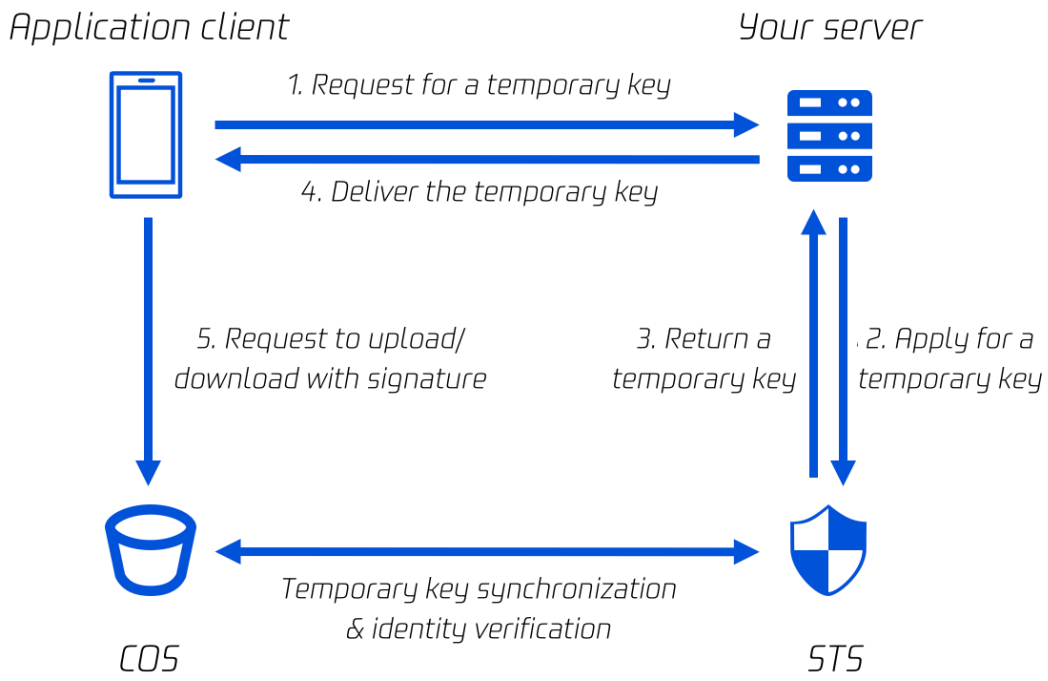
- API 요청 발송
  영구 키와 마찬가지로 임시 키를 통해 서명을 생성하고 요청 헤더에 Authorization을 입력하여 서명된 요청을 형성할 수도 있습니다. COS는 요청을 수신한 후 서명이 유효한지, 임시 키가 만료되었는지 확인합니다.
  서명 알고리즘에 대한 소개는 서명 요청을 참고하십시오. COS는 서명 생성 툴을 제공하며, SDK를 통해서도 서명을 생성할 수 있습니다. SDK 서명 구현을 참고하십시오.
- SDK 툴 사용
  SDK 툴을 설치한 후 임시 키를 사용하여 사용자 신분 정보를 초기화하는 것 외에도 임시 키(SecretId, SecretKey, Token)를 사용하여 COSClient를 초기화하고 서명을 생성하지 않고 SDK를 직접 사용하여 업로드, 다운로드 등을 수행합니다. 임시 키 생성은 임시 키 생성 및 사용 지침을 참고하십시오.

Java SDK 참고 예시는 다음과 같으며 더 많은 언어 demo는 SDK 개요를 참고하십시오.

```
// 1 획득한 임시 키(tmpSecretId, tmpSecretKey, sessionToken) 전송
String tmpSecretId = "SECRETID";
String tmpSecretKey = "SECRETKEY";
String sessionToken = "TOKEN";
BasicSessionCredentials cred = new BasicSessionCredentials(tmpSecretId, tmpSecret
Key, sessionToken);
// 2 bucket의 리전 설정. COS 리전의 약칭은 https://cloud.tencent.com/document/produc
t/436/6224 참고하십시오.
// clientConfig에 region, https(기본값: http), 타임아웃, 프록시 등을 설정하는 set 메소
드가 포함되어 있습니다. 사용 시 소스 코드 또는 FAQ의 Java SDK 부분을 참고하십시오.
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
// 3 cos 클라이언트 생성
COSClient cosClient = new COSClient(cred, clientConfig);
```

임시 키 사용 시나리오

임시 키는 주로 3rd party가 COS에 임시로 액세스할 수 있는 권한을 부여하는 데 사용됩니다. 예를 들어 사용자가 클라이언트 App을 개발하여 COS 버킷에 데이터를 저장하는 경우 이때 영구 키를 App 클라이언트에 직접 저장하는 것은 안전하지 않지만 클라이언트에게 업로드 및 다운로드 권한을 부여해야 하는 이런 시나리오에서 임시 키를 사용할 수 있습니다.



상기 이미지와 같이 사용자가 App 클라이언트를 개발하고 사용자의 서버에 영구 키를 저장합니다. 프론트 엔드 다이렉트 업로드를 위해 임시 키를 사용하려면 다음 단계가 필요합니다.

1. App 클라이언트는 데이터 업로드 및 다운로드를 위해 사용자 서버에 임시 키를 요청합니다.
2. 사용자 서버는 영구 키 신분을 사용하여 STS 서버에서 임시 키를 신청합니다.
3. STS 서비스는 임시 키를 사용자 서버에 반환합니다.
4. 사용자 서버는 임시 키를 App 클라이언트로 전달합니다.
5. App 클라이언트는 임시 키를 사용하여 서명 요청을 생성하고 COS에 데이터 업로드 및 다운로드를 요청합니다.

임시 키는 직접 프런트 엔드 데이터 다이렉트 업로드에 적합하며 다음 모범 사례를 참고하여 임시 키를 사용할 수 있습니다.

- Web의 다이렉트 업로드 사례
- 미니프로그램 다이렉트 업로드 사례

- 모바일 애플리케이션 다이렉트 업로드 사례

사전 서명된 URL을 사용한 COS 액세스

최종 업데이트 날짜: : 2022-12-19 16:13:03

COS(Cloud Object Storage)는 사전 서명된 URL을 사용하여 객체를 업로드 및 다운로드할 수 있도록 지원하며, 서명된 링크를 생성하기 위해 URL에 서명을 삽입하는 것이 원칙입니다. 서명의 유효 기간을 통해 사전 서명된 URL의 유효 기간을 제어할 수 있습니다.

사전 서명된 URL을 사용하여 다운로드할 수 있으며 임시 URL을 가져와 파일, 폴더를 공유할 수 있습니다. 또는 긴 서명 유효 기간을 설정하여 장기간 유효한 URL을 제공 받아 파일을 공유할 수도 있습니다. 세부 사항은 [파일 공유](#파일 공유)를 참고하십시오.

사전 서명된 URL로도 업로드가 가능하며, 자세한 내용은 [파일 업로드](#파일 업로드)를 참고하십시오.

파일 공유(파일 다운로드)

COS는 객체 공유를 지원하며 사전 서명된 URL을 사용하여 제한된 시간 동안 다른 사용자와 파일 및 폴더를 공유할 수 있습니다. 사전 서명된 URL의 원리는 객체 URL 뒤에 서명을 접합하는 것입니다. 서명 생성 알고리즘은 서명 요청을 참고하십시오.

버킷은 기본적으로 개인 읽기이며 객체 URL을 통해 직접 다운로드하면 액세스 실패 메시지가 표시됩니다. 객체 url 뒤에 유효한 서명 접합 후 **사전 서명된 URL**을 얻습니다. 서명은 신분 정보를 전달하므로 사전 서명된 URL을 사용하여 객체를 다운로드할 수 있습니다.

> 설명 :
> 사전 서명을 생성하기 위해 영구 키를 사용해야 하는 경우 위험을 방지하기 위해 영구 키의 권한 범위를 업로드 또는 다운로드 작업으로 제한하는 것이 좋습니다. 그리고 생성된 서명의 유효 기간은 업로드 또는 다운로드 작업을 완료하는 데 필요한 가장 짧은 기간으로 설정합니다. 지정된 사전 서명된 URL의 유효 기간이 만료되면 요청이 중단되기 때문에 새 서명을 신청한 후 실패한 요청은 다시 실행해야 하며, 체크포인트 재시작을 지원하지 않습니다.

```
// 객체 URL
https://test-12345678.cos.ap-beijing.myqcloud.com/test.png
// 사전 서명된 URL(서명 값이 접합된 객체 URL)
https://test-12345678.cos.ap-beijing.myqcloud.com/test.png?q-sign-algorithm=sha1&q-ak=xxxxx&q-sign-time=1638417770;1638421370&q-key-time=1638417770;1638421370&q-header-list=host&q-url-param-list=&q-signaturexxxxxfxxxxxx6&x-cos-security-token=xxxxxxxxxxx
```

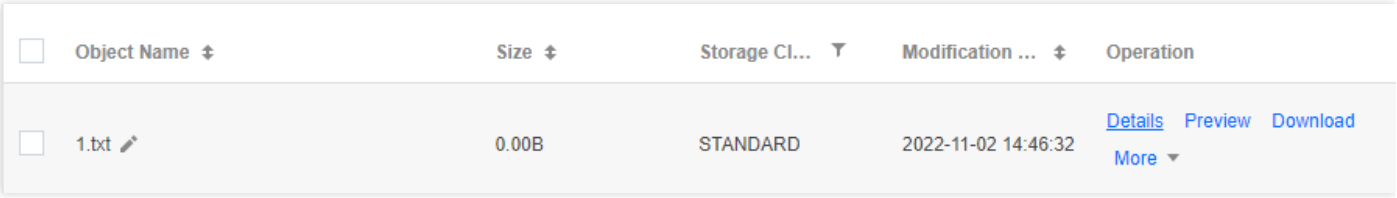
다음 파일 공유 방법은 본질적으로 모두 자동으로 서명을 생성하고 객체 URL 뒤에 접합하며 다운로드 및 미리보기에 직접 사용할 수 있는 임시 링크를 생성합니다.

임시 링크 빠르게 획득(1 - 2시간 동안 유효)

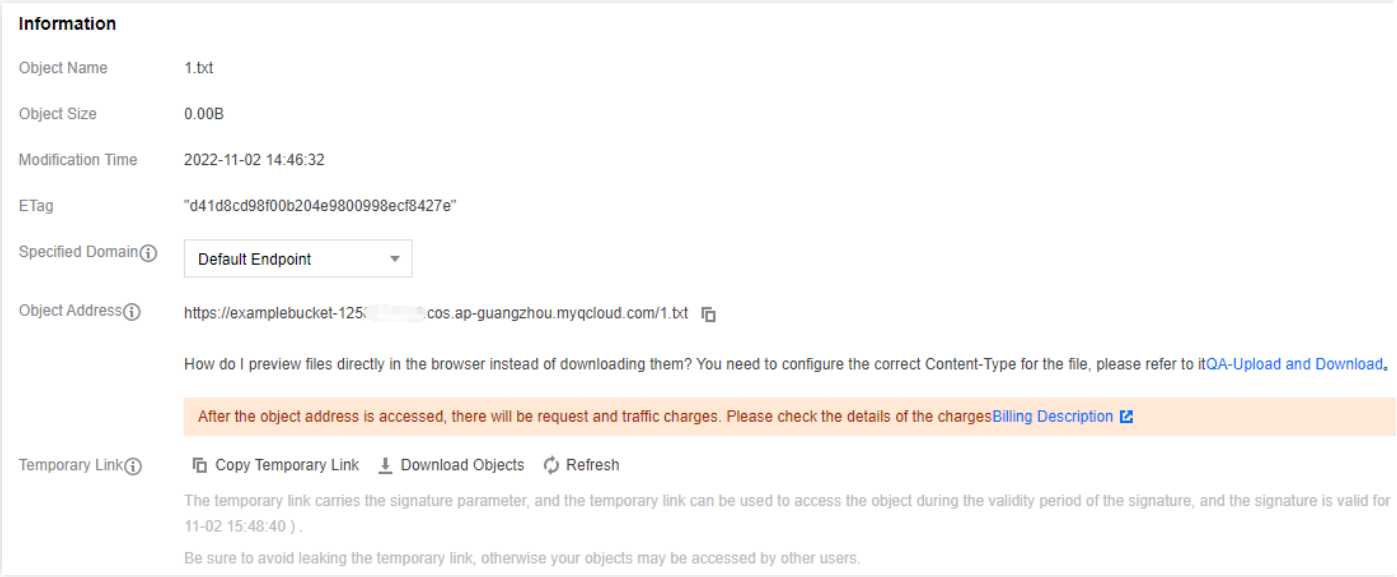
콘솔 혹은 COSBrowser 툴을 통해 객체에 대한 임시 링크를 빠르게 얻을 수 있습니다.

콘솔(Web 페이지)

1. COS 콘솔에 로그인하여 버킷 이름을 클릭하고 '파일 리스트'를 입력한 다음 **세부 사항** 객체를 클릭합니다.



2. 객체 세부 정보 페이지로 이동하여 유효한 1시간 동안 임시 링크를 복사합니다.



COSBrowser(클라이언트)

파일 링크 생성 문서를 참고하여, 루트 계정 키를 사용하여 최대 2시간의 임시 링크를 획득하고, 서브 계정 키를 사용하여 최대 1.5일의 임시 링크를 획득합니다.

사용자 정의 시간의 임시 링크 가져오기

서명 툴 사용

시나리오에 적합: 프로그래밍에 익숙하지 않은 사용자

작업 순서는 다음과 같습니다.

1. 파일 링크: COS 콘솔에 로그인 하여 객체 세부 사항에서 서명 없이 '객체 주소'를 가져옵니다.
2. API Keys 관리에서 SecretId와 SecretKey를 가져옵니다.
3. COS 서명 툴을 클릭하여 서명 링크를 가져옵니다.
   유효 기간: 초, 분, 시 및 일 레벨 설정을 지원합니다.

SDK를 사용하여 일괄적으로 사전 서명된 URL 가져오기

시나리오에 적합: 임시 링크 일괄 가져오기, 프로그래밍 기반의 사용자

콘솔과 COSBrowser에서 얻은 임시 링크는 유효 기간이 짧으며, 더 긴 임시 링크가 필요한 경우 SDK를 사용하여 사전 서명된 URL을 생성할 수도 있습니다. 이는 서명 기간을 제어하여 구현할 수 있습니다. 생성 방법은 사전 서명된 URL을 통해 다운로드를 참고하여 익숙한 개발 언어를 선택하십시오.

임시 키 또는 영구 키를 사용하여 사전 서명된 URL을 생성할 수 있습니다. 둘의 차이점은 임시 키는 최대 유효 기간이 36시간을 넘지 않고 영구 키는 만료되지 않는다는 점으로 이는 사전 서명된 URL의 유효 기간에 간접적인 영향을 미칩니다.

영구 키를 사용하여 사전 서명된 URL 생성(임의 기간)

영구 키는 만료되지 않으며 사전 서명된 URL의 유효 기간은 설정한 서명 유효 기간에 따라 결정됩니다. SDK의 사전 서명된 URL 방법을 직접 호출할 수 있습니다. 작업 단계는 다음과 같습니다.

1. secret_id, secret_key, region 등을 입력하여 client를 초기화합니다.
2. 버킷 이름, 객체 이름 및 서명 유효 기간을 입력하여 사용자 지정 기간으로 사전 서명된 URL을 생성합니다. 자세한 내용은 다음 언어별 SDK 문서를 참고하십시오.

Android SDK	C SDK	C++ SDK	.NET SDK
Go SDK	iOS SDK	Java SDK	JavaScript SDK
Node.js SDK	PHP SDK	Python SDK	미니프로그램 SDK

임시 키로 사전 서명된 URL 생성(36시간 이내)

프런트 엔드 다이렉트 업로드 시나리오에서는 임시 키를 사용해야 하는 경우가 많습니다. 임시 키에 대한 설명 및 생성 가이드는 다음을 참고하십시오.

- 임시 키를 사용한 COS 액세스
- 임시 키 생성 및 사용 가이드
- 프런트엔드에서 COS로 다이렉트 업로드 시 임시 자격 증명 사용 보안 가이드

임시 키의 최대 유효 기간은 36시간이며, 사전 서명된 URL의 유효 기간은 귀하가 설정한 서명의 유효 기간과 임시 키의 유효 기간의 최소값입니다. 서명의 유효 기간을 X로 설정하고 임시 키의 유효 기간을 Y로, 링크의 실제 유효 시간을 T라고 가정합니다.

```
T=min(X,Y); X<=36 이므로 T<=36.
```

임시 키로 사전 서명된 URL을 생성하려면 다음 두 단계가 필요합니다.

1. 임시 키 획득.
2. 임시 키 획득 후 영구 키와 유사한 함수를 사용하여 사전 서명된 URL을 생성할 수 있습니다. 임시 키로 client를 초기화하기 위해서는 SecretId, SecretKey 뿐만 아니라 token도 입력해야 하며 'x-cos-security-token' 매개변수가 전달된다는 점에 유의해야 합니다. 자세한 내용은 다음 언어별 SDK 문서를 참고하십시오.

Android SDK	C SDK	C++ SDK	.NET SDK
Go SDK	iOS SDK	Java SDK	JavaScript SDK
Node.js SDK	PHP SDK	Python SDK	미니프로그램 SDK

폴더 공유

폴더는 특별한 객체이며 콘솔 또는 COSBrowser 툴을 통해 폴더를 공유할 수 있습니다. 자세한 내용은 폴더 공유를 참고하십시오.

파일 업로드

기본적으로 버킷과 객체는 개인 소유입니다. 3rd party가 버킷에 객체를 업로드할 수 있지만 CAM 계정 혹은 임시 키 사용을 원치 않는 경우 임시 업로드 작업을 완료하기 위해 URL 사전 서명을 사용해 3rd party에게 서명을 제출합니다. 유효한 서명 URL을 받은 계정은 모두 객체 업로드를 진행할 수 있습니다.

> 설명 :
> 사전 서명을 생성하기 위해 영구 키를 사용해야 하는 경우 위험을 방지하기 위해 영구 키의 권한 범위를 업로드 또는 다운로드 작업으로 제한하는 것이 좋습니다. 그리고 생성된 서명의 유효 기간은 업로드 또는 다운로드 작업을 완료하는 데 필요한 가장 짧은 기간으로 설정합니다. 지정된 사전 서명된 URL의 유효 기간이 만료되면 요청이 중단되기 때문에 새 서명을 신청한 후 실패한 요청은 다시 실행해야 하며, 체크포인트 재시작을 지원하지 않습니다.

- 방법1: SDK를 사용하여 사전 서명된 URL 생성
  각 언어 SDK는 사전 서명된 URL을 생성하고 업로드하는 방법을 제공하며, 생성 방법은 사전 서명된 URL을 통해 업로드를 참고하여 익숙한 개발 언어를 선택하십시오.

- 방법2: 서명 링크 자체 접합
  사전 서명된 URL은 실제로 객체 URL 뒤에 접합된 서명이므로 SDK, 서명 생성 툴 등을 통해 직접 서명을 생성하고 객체 업로드를 위해 URL과 서명을 서명된 링크에 접합합니다. 그러나 서명 생성 알고리즘의 복잡성으로 인해 이 사용 방법은 일반적으로 권장되지 않습니다.

익명 COS 액세스

최종 업데이트 날짜: : 2022-03-09 17:26:07

COS(Cloud Object Storage) 버킷은 기본적으로 비공개이며 COS에 대한 액세스는 인증되어야 하며 객체 URL을 통한 COS에 대한 액세스에는 서명이 있어야 합니다. 그러나 리소스(버킷, 객체, 폴더)가 공개 읽기로 개방되면 익명 액세스가 허용됩니다. 즉, 객체 URL을 통해 리소스를 직접 다운로드할 수 있습니다.

공개 권한 범위에 따라 COS는 버킷 레벨, 객체 레벨, 폴더 레벨에서 공개 읽기 설정을 지원합니다.

버킷을 공개 읽기로 개방

버킷을 공개 읽기/비공개 쓰기로 개방하면 버킷의 모든 객체는 익명으로 액세스될 수 있습니다. 설정 방법은 버킷 액세스 권한 설정을 참고하십시오.

객체를 공개 읽기로 개방

공개 읽기/비공개 쓰기로 지정된 객체를 개방하면 객체 URL을 통해 객체에 직접 액세스할 수 있습니다. 설정 방법은 객체 액세스 권한 설정을 참고하십시오.

폴더를 공개 읽기로 개방

공개 읽기/비공개 쓰기로 폴더를 개방하면 폴더의 모든 파일은 익명으로 액세스될 수 있습니다. 설정 방법은 폴더 권한 설정을 참고하십시오.

공개 읽기 권한 평가 메커니즘

COS 권한 평가 메커니즘은 액세스 정책 평가 프로세스를 참고십시오. 버킷, 폴더, 객체 레벨에서 공개 읽기 권한 설정이 충돌할 경우 우선순위 정렬은 다음과 같습니다.

객체 ACL에 대해 객체 ACL이 가장 높은 우선순위를 가지며, 객체 ACL이 상속된 권한이면 폴더 ACL이 우선하고, 폴더 ACL이 상속된 권한이면 버킷 ACL이 우선합니다.

CDN 가속으로 액세스
CDN 가속 개요

최종 업데이트 날짜: : 2023-03-14 14:48:45

Content Delivery Network(CDN)을 사용하여 Cloud Object Storage(COS) 버킷에 저장된 콘텐츠를 대량 다운로드/전송할 수 있으며, 이는 동일한 콘텐츠를 반복적으로 다운로드해야 하는 경우에 이상적입니다. Origin-pull 인증 기능을 통해 CDN은 개인 읽기 버킷에 저장된 콘텐츠를 가속화할 수 있습니다. CDN 인증 기능은 데이터 보안 위험과 불필요한 트래픽 비용 발생을 방지하기 위해 승인된 사용자만 콘텐츠를 다운로드할 수 있습니다.

> 설명 :
> 사용자가 CDN 가속 도메인 이름을 활성화한 후 CDN 가속 도메인 이름을 사용하여 데이터를 다운로드하고 액세스하면 CDN Origin-pull 트래픽과 CDN 트래픽이 생성됩니다. 자세한 내용은 COS를 CDN 원본 서버로 설정 시 발생하는 트래픽을 참고하십시오.

CDN

CDN의 정의

CDN은 전 세계에 분산된 고성능 엣지 노드로 구성된 Internet 에코시스템 레이어입니다. 이러한 노드는 캐시 정책에 따라 콘텐츠를 저장합니다. 사용자가 콘텐츠를 요청하면 액세스 속도를 높이고 가용성을 향상시키기 위해 요청이 사용자와 가장 가까운 엣지 노드로 라우팅됩니다.

CDN에는 캐시 및 Origin-pull이 포함됩니다. 사용자가 URL에 액세스할 때 요청한 콘텐츠가 엣지 노드에 캐시되지 않았거나 캐시된 콘텐츠가 만료된 경우 해당 콘텐츠를 원본에서 가져옵니다.

적용 시나리오

- 응답 지연 및 다운로드 속도에 대한 요구 사항이 높은 시나리오.
- 지역, 국가, 대륙 간 GB ~ TB 용량의 데이터를 전송하는 시나리오
- 같은 콘텐츠를 짧은 시간에 반복적으로 다운로드하는 시나리오.

보안 유형

- Origin-pull 인증: 요청한 콘텐츠가 엣지 노드에 캐시되지 않은 경우 콘텐츠를 원본에서 가져옵니다. COS가 원본으로 사용되고 Origin-pull 인증이 활성화된 경우 CDN 엣지 노드는 특수 서비스 ID를 사용하여 COS 원본에 액세스하여 프라이빗 버킷의 데이터를 획득하고 캐시합니다.

- CDN 서비스 인증: CDN 엣지 노드가 특수 서비스 ID를 사용하여 COS 원본에 액세스하고 콘텐츠를 가져올 수 있도록 CDN 서비스를 인증할 수 있습니다.
- CDN 인증: 사용자가 엣지 노드에 액세스하여 캐시 데이터를 획득하면 엣지 노드는 인증 구성 규칙에 따라 액세스 URL의 인증 필드를 확인합니다. 이를 통해 무단 액세스 및 링크 도용을 방지하여 엣지 노드에 캐시된 데이터의 보안과 안정성을 향상시킵니다.

COS 액세스 노드

액세스 노드의 정의

액세스 노드는 버킷의 리전 및 이름에 따라 버킷에 할당되는 도메인 이름입니다. 이 도메인 이름을 사용하여 버킷에 저장된 데이터에 액세스할 수 있습니다.

정적 웹사이트 기능이 활성화되면 기본 액세스 노드와 다른 응답을 구성할 수 있는 정적 웹사이트 액세스 노드가 제공됩니다.

액세스 노드

- 액세스 노드: 버킷이 생성되면 COS는 RESTful API를 사용하여 액세스할 수 있는 <BucketName-APPID>.cos.<Region>.myqcloud.com 형식의 XML 액세스 노드를 버킷에 할당합니다. 노드에 액세스하고 API 문서를 참고하여 버킷을 구성하거나 객체를 업로드/다운로드할 수 있습니다.
- 정적 웹 사이트 노드: 콘솔의 버킷에 대한 기본 구성 페이지에서 정적 웹 사이트 기능을 활성화할 수 있습니다. 그런 다음 <BucketName-APPID>.cos-website.<Region>.myqcloud.com 형식의 액세스 노드가 제공됩니다. 객체 다운로드만 허용하는 정적 웹 사이트에 대한 특수 인덱스 페이지(IndexPage), 오류 페이지(ErrorPage) 및 리디렉션을 구성할 수 있습니다. 사용자는 정적 웹 사이트 도메인 이름을 사용하여 콘텐츠를 얻을 수 있습니다.

액세스 권한

- 공개 읽기: 버킷이 공개 읽기로 설정되어 있으면 누구나 해당 도메인 이름을 사용하여 버킷에 액세스할 수 있습니다. 공개 읽기 버킷을 원본으로 사용하는 경우 CDN 인증 및 Origin-pull 인증을 활성화하지 않고도 CDN 가속을 직접 활성화할 수 있습니다.
- 개인 읽기: 버킷이 개인 읽기로 설정된 경우 액세스 정책을 생성하여 버킷에 액세스할 수 있는 사람을 관리하고 CDN 권한 부여를 관리할 수 있습니다. 개인 읽기 버킷을 원본으로 사용하고 Origin-pull 인증을 활성화하지만 CDN 인증은 활성화하지 않으면 권한이 없는 사용자가 CDN을 통해 버킷에 액세스할 수 있습니다. 따라서 **데이터 보안을 보장하기 위해 개인 읽기 버킷에 대해 Origin-pull 인증과 CDN 인증을 모두 활성화하는 것이 좋습니다**.

CDN을 사용하여 COS 액세스 가속화

ICP 비안 번호를 획득한 사용자 지정 도메인 이름을 사용하고 COS 버킷을 오리진으로 사용할 수 있습니다. 이렇게 하면 사용자 지정 CDN 가속 도메인 이름을 사용하여 버킷의 객체에 대한 액세스를 가속화할 수 있습니다.

설명：
Tencent Cloud CDN 가속 도메인 이름은 기본적으로 IP 주소를 제공하지 않습니다. 도메인 이름의 DNS에 대해 자세히 알아보려면 쿼리를 위해 dig 명령을 실행하십시오.

공개 읽기 버킷

버킷이 공개 읽기로 설정되고 COS가 CDN 풀링의 원본으로 사용되는 경우 Origin-pull 인증을 활성화할 필요가 없으며 CDN 엣지 노드는 COS 버킷에 저장된 객체를 가져와 캐시할 수 있습니다.

CDN 콘솔에서 인증 설정을 활성화하여 버킷의 객체를 **어느 정도** 보호할 수 있습니다. 이 기능의 활성화 여부에 관계없이 버킷 액세스 도메인 이름을 아는 사용자는 버킷의 모든 객체에 액세스할 수 있습니다. 사용자가 다양한 CDN 인증 구성에서 공개 읽기 버킷에 액세스할 수 있는지 여부는 다음 표에 설명되어 있습니다.

CDN 인증	CDN 가속 도메인 이름으로 액세스	COS 도메인 이름으로 액세스	사용 사례
비활성화(디폴트)	액세스 가능	액세스 가능	CDN 또는 원본을 통해 전체 웹사이트에 대한 모든 공개 액세스 허용
활성화	URL 인증 필요	액세스 가능	원본이 아닌 CDN을 통한 액세스에 대해 링크 도용 방지 활성화(권장하지 않음)

개인 읽기 버킷

버킷이 개인 읽기(기본값)로 설정되고 COS가 CDN 풀링의 원본으로 사용되는 경우 CDN 엣지 노드는 **객체를 가져오고 캐시할 수 없습니다**. 따라서 버킷 정책(Bucket Policy)에 CDN 서비스 ID를 추가하고 ID에 다음 작업을 수행할 수 있는 권한을 부여해야 합니다.

- GET Object: 객체 다운로드
- HEAD Object: 객체 메타데이터 쿼리
- OPTIONS Object: CORS 실행 전 요청 구성

CDN 서비스 인증 추가를 클릭하면 CDN 콘솔 또는 COS 콘솔에서 원클릭으로 ID를 인증할 수 있습니다. 그 다음 **Origin-pull 인증**을 활성화합니다. 이러한 방식으로 CDN 엣지 노드는 서비스 ID를 사용하여 COS 객체에 액세스할 수 있습니다.

주의：

- 버킷이 개인 읽기로 설정된 경우 CDN 서비스 권한 부여를 추가하고 origin-pull 인증을 활성화해야 합니다. 그렇지 않으면 COS에 대한 액세스가 거부됩니다.

- CDN 엣지 노드는 각 루트 계정에 대한 서비스 계정을 생성합니다. 따라서 계정 인증은 가속 도메인 이름이 속한 루트 계정에 대해서만 유효합니다. 가속 도메인 이름이 다른 계정에 바인딩된 경우 액세스가 거부됩니다.

CDN 서비스 권한 부여를 추가하고 Origin-pull 인증을 활성화하면 CDN 엣지 노드가 데이터를 가져와 캐시할 수 있습니다. 따라서 버킷에 저장된 개인 데이터를 보호해야 하는 경우 인증 설정을 활성화하는 것이 좋습니다. 다양한 CDN 인증 구성에 대한 개인 읽기 버킷 액세스 가능 여부는 다음 표에 설명되어 있습니다.

CDN 인증	CDN 가속 도메인 이름으로 액세스	COS 도메인 이름으로 액세스	사용 사례
비활성화 (디폴트)	액세스 가능	COS 인증 필요	원본의 콘텐츠를 보호하기 위해 CDN 도메인 이름에 대한 직접 액세스 허용
활성화	URL 인증 필요	COS 인증 필요	포괄적인 액세스 보안(CDN 인증을 위한 링크 도용 방지 지원)

CDN 가속 구성

최종 업데이트 날짜: : 2023-01-13 14:24:23

적용 시나리오

- 응답 지연 및 다운로드 속도에 대한 요구 사항이 높은 시나리오.
- 리전, 국가, 대륙 간에 GB~TB 용량의 데이터를 전송하는 시나리오.
- 같은 콘텐츠를 짧은 시간에 반복적으로 다운로드하는 시나리오.

> 주의 :
> 다운로드 요청 출처가 Tencent Cloud VPC인 경우(예: CVM을 사용하여 COS 버킷에 액세스) COS 도메인을 직접 사용하는 것이 좋습니다. 사용자 지정 CDN 가속 도메인을 사용하는 경우 공중망을 통해 CDN 노드에 액세스해야 하며, 이때 CDN Origin-pull 요금 및 CDN 트래픽 요금과 같은 추가 요금이 발생합니다.

관련 설명

도메인의 정의, CDN Origin-pull 인증, CDN 인증 설정에 대한 내용은 CDN 가속 개요 문서의 소개 내용을 확인하십시오.

CDN Origin-pull 인증, CDN 인증 구성은 사용자 지정 CDN 가속 도메인 이름 및 COS 도메인 이름이 원본 버킷에 액세스하는 방식에 영향을 미칩니다. 자세한 내용은 다음 표를 참고하십시오.

버킷 액세스 권한	CDN Origin-pull 인증	CDN 인증	사용자 지정 CDN 가속 도메인 이름을 통해 원본에 액세스 가능	COS 엔드포인트를 통해 원본에 액세스 가능	시나리오
공개 읽기	비활성화	비활성화	Yes	Yes	전체 서버 공용 액세스
공개 읽기	활성화	비활성화	Yes	Yes	비권장
공개 읽기	비활성화	활성화	URL 인증 필요	Yes	비권장
공개 읽기	활성화	활성화	URL 인증 필요	Yes	비권장

버킷 액세스 권한	CDN Origin-pull 인증	CDN 인증	사용자 지정 CDN 가속 도메인 이름을 통해 원본에 액세스 가능	COS 엔드포인트를 통해 원본에 액세스 가능	시나리오
개인 읽기+CDN 서비스 라이선스	활성화	활성화	URL 인증 필요	COS 인증 필요	전체 링크 보호
개인 읽기+CDN 서비스 라이선스	비활성화	활성화	URL 인증 필요	COS 인증 필요	비권장
개인 읽기+CDN 서비스 라이선스	활성화	비활성화	Yes	COS 인증 필요	원본 보호
개인 읽기+CDN 서비스 라이선스	비활성화	비활성화	No	COS 인증 필요	비권장
개인 읽기	비활성화	활성화 또는 비활성화	No	COS 인증 필요	CDN 이용 불가

주의 :
위의 표에 기재된 **원본 서버 보호** 시나리오의 경우, CDN 인증 설정을 활성화하지 않으면 CDN 엣지 노드에 캐싱된 데이터가 악의적으로 풀링될 수 있으니 CDN 인증 설정을 활성화하여 데이터 보안을 강화하시기 바랍니다.

CDN 가속 설정

COS 콘솔에서 버킷에 사용자 정의 도메인을 바인딩한 다음 사용자 정의 도메인 CDN 가속을 활성화하고, 마지막으로 사용자 정의 도메인 가속을 통해 메모리 버킷에 액세스할 수 있습니다. 사용자 정의 도메인을 바인딩할 때 사용자 정의 도메인의 서비스 공급자로부터 CNAME 리졸브를 직접 추가해야 합니다.

주의 :
현재 COS에서 사용자 정의 가속 도메인을 사용할 경우 반드시 CDN을 활성화해야 합니다. 실제 상황에 따라 선택하여 적용하시기 바랍니다.

1. 도메인으로 중국 내 CDN에 액세스하는 경우, ICP비안이 필요합니다. 하지만 Tencent Cloud의 경우, ICP비안이 필수사항이 아니며, 액세스할 도메인이 등록된 도메인이면 가능합니다.
2. 도메인으로 해외 CDN에 액세스하는 경우, ICP비안이 필요하지 않습니다. 단, Tencent Cloud에 저장한 데이터와 작업에 관해서는 관련 국가의 법률, 법규 및 를 준수해야 합니다.

전제 조건

1. 도메인을 등록합니다. 도메인은 Tencent Cloud 도메인 등록 또는 기타 서비스 제공 업체를 통해 등록할 수 있습니다.
2. ICP비안 신청서. 중국 본토에서 콘텐츠를 제공하려면 ICP비안을 완료하십시오.

작업 단계

> 주의 :
> COS 콘솔과 CDN 콘솔에서 사용자 정의 도메인을 추가할 수 있고, CDN 가속 활성화도 가능합니다. CDN 콘솔에서 사용자 정의 도메인을 추가하려면 도메인 연결을 참고하십시오.

1. 버킷 선택

COS 콘솔에 로그인한 뒤 왼쪽 사이드바에서 **버킷 리스트**를 클릭하고, CDN 가속을 활성화할 버킷 이름을 클릭합니다.

2. 사용자 정의 CDN 가속 도메인 추가

도메인 및 전송 관리 > 사용자 지정 CDN 가속 도메인을 클릭하고 **사용자 정의 CDN 가속 도메인** 구성 항목 아래에서 **도메인 추가**를 클릭하여 구성 가능한 상태로 들어갑니다.

- **도메인 이름**: 타깃 사용자 정의 도메인 이름(예시: `www.example.com`)을 입력합니다. ICP 비안이 완료되고 CNAME 레코드가 입력한 도메인에 대한 DNS 서비스 제공 업체에서 구성되었는지 확인하십시오. 자세한 내용은 CNAME 설정을 참고하십시오. 연결하려는 사용자 정의 CDN 가속 도메인이 다음과 같은 상황에 있는 경우 도메인 소유권 인증의 지침에 따라 도메인 소유권을 확인해야 합니다.
  - 처음으로 연결하는 도메인
  - 다른 사용자에 의해 연결된 도메인
  - 와일드카드 서브도메인
- **가속 리전**: CDN 가속은 중국 내, 중국 외 및 전 세계 리전에 대해 지원되며 모든 리전의 버킷에 대한 글로벌 가속이 있습니다.
- **원본 서버 유형**: 기본값은 **기본 원본 서버**입니다. 원본 서버인 버킷으로 정적 웹 사이트를 활성화한 뒤 가속하려면 원본 서버 유형을 **정적 웹 사이트 원본 서버**로 설정하십시오. 자세한 내용은 CDN 가속 개요를 참고하십시오.
- **인증**: Origin-pull 인증을 활성화합니다. 개인 읽기 버킷의 경우, **Origin-pull 인증을 활성화**하여 원본 서버를 보호합니다.

> 주의 :
>
> 개인 읽기 버킷의 경우 Origin-Pull 인증과 CDN 서비스 권한이 모두 활성화되어 있으면 CDN을 통해 원본에 액세스하는 데 서명이 필요하지 않으며 CDN에 캐시된 리소스가 공중망에 배포되므로 데이터 보안에 영향을 미칩니다. 따라서 CDN 인증을 활성화하는 것이 좋습니다.

- Origin-pull 인증 활성화
  Origin-pull 인증은 불법 액세스를 차단하기 위해 CDN 엣지 노드의 서비스 자격을 인증하는 용도입니다. 자세한 내용은 다음과 같습니다.
  - 공개 읽기 버킷: 라이선스가 없어도 CDN 엣지 노드에서 버킷에 바로 액세스할 수 있으므로 Origin-pull 인증을 실행하지 않아도 됩니다.
  - 개인 읽기 버킷: CDN 엣지 노드는 Origin-pull 인증을 통해 서비스 자격을 인증해야 하며, 인증을 통과해야 버킷에 저장된 객체에 액세스할 수 있습니다. **Origin-pull 인증** 활성화를 선택하십시오.

Origin-pull 인증 활성화 후, 오른쪽의 **저장**을 클릭하고 5분 정도 기다리면 사용자 정의 도메인이 추가되고, CDN 가속 배포가 완료됩니다.

- CDN 인증 활성화

> 주의 :
>
> 사용자 지정 도메인 이름에 대해 CDN 가속이 활성화되면 누구나 도메인 이름을 통해 원본 서버에 액세스할 수 있습니다. 따라서 데이터를 비공개로 유지해야 하는 경우 CDN 인증을 활성화하여 원본 서버에서 데이터를 보호해야 합니다.

  사용자 정의 도메인의 배포가 완료되면 CDN 인증 메뉴에 CDN 인증 기능 설정을 위한 링크가 나타납니다. 이때 **설정**을 클릭하면 CDN 콘솔로 이동해 CDN 인증을 설정할 수 있습니다. 자세한 내용은 인증 설정을 참고하십시오.

- **CDN 캐시 자동 퍼지**: 활성화되면 COS 버킷 업데이트 파일 규칙이 트리거될 때 CDN 캐시가 자동으로 퍼지됩니다. COS 함수 계산 설정으로 이동할 수 있습니다. 작업 가이드는 CDN 캐시 퍼지 설정을 참고하십시오.

- **HTTPS 인증서**: 사용자 정의 CDN 가속 도메인에 HTTPS 인증서를 추가하려면 CDN 콘솔로 이동하여 설정할 수 있습니다.

3. 도메인 이름 리졸브

사용자 지정 도메인 이름이 CDN에 추가되면 시스템에서 '.cdn.dnsv1.com' 접미사가 붙은 CNAME 도메인 이름을 자동으로 할당합니다. 도메인 이름 서비스 공급자에서 CNAME 구성을 완료해야 합니다. 자세한 내용은 CNAME 설정을 참고하십시오.

> 주의 :
> CNAME 도메인으로 바로 액세스할 수 없습니다.

4. 기능 비활성화

위의 단계가 완료되면 사용자 정의 도메인을 통해 버킷 내 리소스에 대한 액세스 를 가속화할 수 있습니다. 사용자 지정 CDN 가속을 비활성화 하려면 다음과 같은 방법으로 비활성화할 수 있습니다.

사용자 정의 CDN 가속 도메인 관리 인터페이스에서 **편집**을 클릭하여 도메인 상태를 변경할 수 있습니다. 도메인 상태를 **사용 중**에서 **사용하지 않음**으로 변경한 뒤 **저장**을 클릭하여 배포합니다. 약 5분 경과 후 기능이 비활성화되며, CDN 콘솔의 해당 사용자 정의 가속 도메인 상태도 **활성화됨**에서 **비활성화됨**으로 변경됩니다.

> 주의 :
> 사용자 정의 도메인을 삭제할 경우, 사용자 정의 도메인이 **사용 중** 상태일 때는 바로 삭제할 수 없습니다. 상태를 **사용하지 않음**으로 변경한 뒤 삭제하십시오. CDN 콘솔에서 도메인의 비활성화 또는 삭제 작업을 실행할 수 있으며, 자세한 내용은 도메인 작업을 참고하십시오.

단일 링크 속도 제한

최종 업데이트 날짜: : 2021-03-02 10:38:38

단일 링크 속도 제한

COS는 다른 애플리케이션의 네트워크 대역폭을 고려해 파일을 업로드 및 다운로드할 때 발생하는 트래픽을 제어합니다. PutObject, PostObject, GetObject, UploadPart에서 요청 시, x-cos-traffic-limit 매개변수를 포함하고, 속도값을 제한하면 COS는 설정한 속도 제한값에 따라 요청한 네트워크 대역폭을 제어합니다.

사용 설명

- 사용자는 PUT Object, POST Object, GET Object, Upload Part 요청을 실행할 때, x-cos-traffic-limit 요청 헤더(POST Object 요청 시, 테이블 필드 요청)를 포함해 해당 요청의 속도 제한값을 설정합니다. 해당 매개변수는 header와 요청 매개변수에 설정하거나 form 업로드 인터페이스를 사용할 경우, form field에서 설정할 수 있습니다.
- x-cos-traffic-limit 매개변수값은 숫자여야 하며, 기본 단위는 bit/s입니다.
- 속도 제한값은 819200 ~ 838860800, 즉 100KB/s ~ 100MB/s 범위에서 설정할 수 있으며, 이 범위를 넘으면 400 오류가 반환됩니다.

> **설명 :**
>
> 단위 환산법: 1MByte=1024KByte=1048576Byte=8388608bit

API 사용 예시

다음은 API의 간편 업로드 예시입니다. 속도 제한값은 1048576 bit/s으로 128KB/s입니다.

```
PUT /exampleobject HTTP/1.1
Host: examplebucket-1250000000.cos.ap-beijing.myqcloud.com
Content-Length: 13
Authorization: q-sign-algorithm=sha1&q-ak=AKID8A0fBVtYFrNm02oY1g1JQQF0c3JO****&q-sign-time=1561109068;1561116268&q-key-time=1561109068;1561116268&q-header-list=content-length;content-md5;content-type;date;host&q-url-param-list=&q-signature=998bfc8836fc205d09e455c14e3d7e623bd2****
x-cos-traffic-limit: 1048576
```

일괄 프로세스
일괄 프로세스 작업 관리

최종 업데이트 날짜: : 2022-05-16 16:42:01

콘솔을 통해 일괄 프로세스 작업을 관리할 수 있으며, 본 문서에서는 일괄 프로세스 작업과 관련한 관리 작업에 대해 자세히 소개합니다.

필터링 작업

List Jobs API를 통해 최근 90일 이내에 생성한 일괄 프로세스 작업을 확인할 수 있습니다. 일괄 프로세스 작업 리스트에는 작업 ID, 작업 설명, 작업 우선순위, 작업 상태, 작업 실행 상황 등 모든 일괄 프로세스 작업 정보가 포함되어 있으며, 작업 상태를 통해 일괄 프로세스 작업 리스트에서 동일한 상태의 작업을 필터링할 수 있습니다. 또한 콘솔에서 필터링 작업 시에는 작업 설명 또는 작업 ID를 통해 필터링할 수도 있습니다.

작업 상태 조회

작업 관련 더 많은 정보는 DescribeJob API를 통해 단일 업무에 대한 모든 정보를 확인할 수 있습니다. 해당 인터페이스는 지정한 작업의 조작 설정, 객체 리스트 정보, 작업 보고서 등의 정보를 반환하며, 이를 통해 지정한 작업에 대한 자세한 정보를 확인할 수 있습니다.

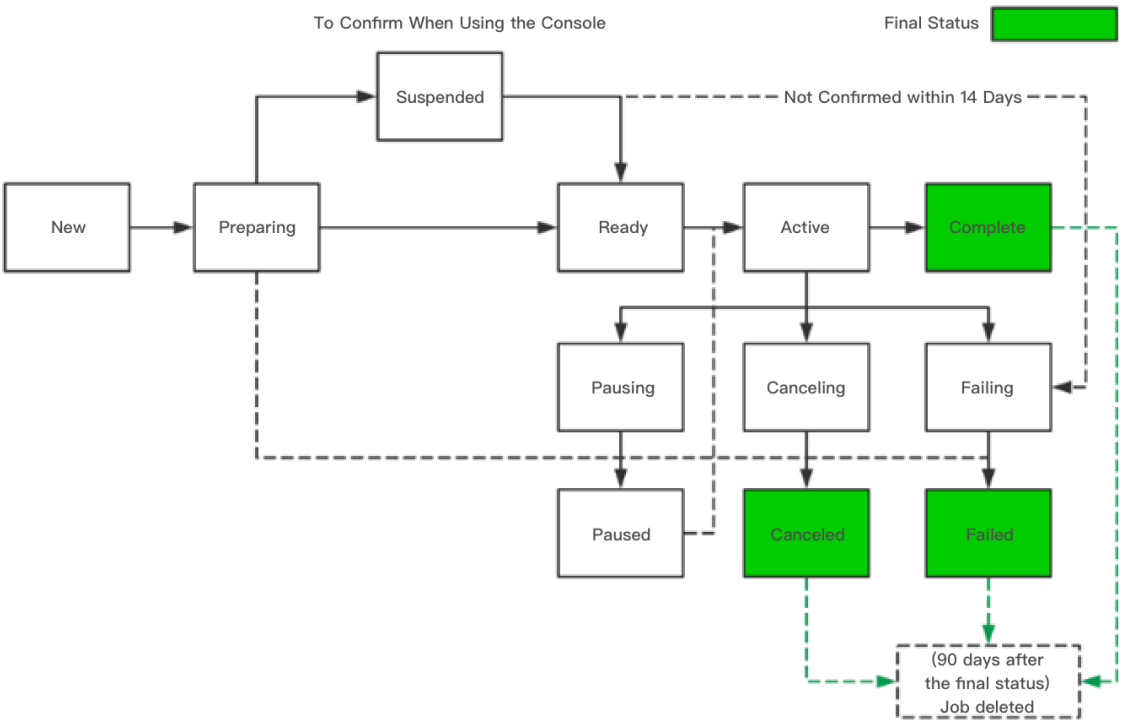
작업 할당 우선순위

작업에 우선순위를 설정할 수 있습니다. COS는 설정한 우선순위에 따라 일괄 프로세스 작업을 진행합니다. 우선순위가 높은 작업은 우선 처리되며, 우선순위는 정수값으로 수치를 표시하고 내림차순으로 정렬되어 있습니다. 즉, 숫자가 클수록 우선순위가 높다는 뜻입니다. 작업 실행 중 우선순위를 수정할 수 있으며, 작업 실행 중에 우선순위가 더 높은 작업을 추가하는 경우, 우선순위가 낮은 작업을 일시 정지하고 우선순위가 높은 작업을 먼저 실행합니다.

> 주의 :
> 통상적으로 우선순위가 높은 일괄 프로세스 작업을 우선순위가 낮은 작업보다 먼지 실행하지만, 우선순위가 순차 작업의 표준이 되지는 없습니다. 여러 항목의 일괄 프로세스 작업을 순차적으로 실행해야 하는 경우 각 작업의 실행 상태를 모니터링하여 작업 상태를 확인하시기 바랍니다.

작업 상태

작업을 생성하면 아래 표의 작업 상태로 전환됩니다. 상태 전환 프로세스는 다음 이미지와 같습니다.



작업 상태에 대한 자세한 의미는 다음 표와 같습니다.

작업 상태	설명	다음 상태
New	작업이 막 생성된 경우 New로 표시됩니다.	Preparing으로 전환될 수 있으며, COS에서 작업 상의 리스트의 리졸브 작업을 시작했다는 의미입니다.
Preparing	COS에서 일괄 프로세스 작업에 설정된 리스트와 기타 정보에 대해 리졸브를 시작하면 Preparing 상태가 됩니다.	Ready 상태 또는 Suspended 상태로 전환될 수 있습니다. Ready 상태로 전환되면 일괄 프로세스 작업 설정 정보를 COS에서 리졸브를 완료했다는 의미입니다. 즉, COS가 설정에 따라 리스트의 객체에 대해 지정 작업을 실행할 예정이라는 의미입니다. Suspended 상태로 전환되면 일괄 프로세스 작업 설정 정보를 COS에서 리졸브를 완료했으나, 귀하가 확인을 해야만 실행할 수 있다는 의미입니다. 이 상태는 콘솔에서 일괄 프로세스 작업을 설정하는 경우 나타납니다.

작업 상태	설명	다음 상태
Suspended	일괄 프로세스 작업이 귀하가 확인을 해야만 실행할 수 있을 경우 Suspended 상태가 됩니다. 이 상태는 콘솔에서 작업 생성 시 발생합니다. 귀하가 확인하고 COS에서 작업을 실행하면 다음 상태로 전환됩니다.	귀하가 작업 실행을 확인하면 Ready 상태로 전환됩니다.
Ready	COS에서 일괄 프로세스 작업의 객체 리스트와 작업 설정 정보에 대해 리졸브를 완료하고 곧 일괄 프로세스 작업을 실행할 때 Ready 상태가 됩니다.	Active 상태로 전환될 수 있으며, 이 때 COS에서 작업을 시작합니다. 우선순위가 더 높은 작업이 현재 실행 중인 경우 COS는 높은 우선순위 작업 상태가 Complete이 될 때까지 작업 상태를 Ready로 유지합니다.
Active	COS 일괄 프로세스 작업에 설정한 정보에 따라 리스트 상의 객체를 작업 중인 경우 Active 상태가 됩니다. 콘솔 또는 DescribeJob API 호출 등의 방법을 통해 진행 상태를 확인할 수 있습니다.	Complete, Failing, Pausing 또는 Cancelling 등의 상태로 전환될 수 있습니다. 작업이 성공적으로 종료되거나 실패할 경우, 또는 귀하의 중지 명령(예: 작업 취소) 등 구체적인 원인에 따라 상태가 전환됩니다.
Pausing	COS에서 현재 처리하고 있는 일괄 프로세스 작업을 중단하고 있는 상태로, Paused로 진행되는 상태가 Pausing 상태입니다.	Paused 상태로 전환됩니다. COS에서 실행 중인 일괄 프로세스 작업을 모두 중지한 경우 Paused 상태로 전환됩니다.
Paused	더 높은 우선순위의 일괄 프로세스 작업을 생성할 경우, 현재 실행 중인 일괄 프로세스 작업이 Paused 상태로 전환됩니다.	우선순위가 높은 작업이 완료, 실패 또는 확인 대기 상태가 되면 Paused 상태인 작업이 자동으로 Active 상태로 전환됩니다.

작업 상태	설명	다음 상태
Complete	일괄 프로세스 작업의 모든 리스트 내 객체의 일괄 작업을 완료하였거나 실패하는 경우 Complete 상태로 전환됩니다. 작업 보고서 생성을 설정한 경우 COS에서 작업 상태를 Complete으로 전환 시 지정한 버킷에 작업 보고서를 전달합니다.	Complete 상태는 마지막 상태로, Complete 상태로 전환되면 다시 다른 상태로 전환되지 않습니다.
Cancelling	COS가 현재 처리하고 있는 일괄 프로세스 작업을 취소하는 중으로, Cancelled로 진행되는 상태가 Cancelling 상태입니다.	Cancelled 상태로 전환됩니다. COS에서 실행 중인 일괄 프로세스 작업을 모두 취소한 경우 Cancelled 상태로 전환됩니다.
Cancelled	일괄 프로세스 작업의 취소 처리가 완료되면 Cancelled 상태로 전환됩니다. 이 때에는 작업 상태를 수정할 수 없습니다.	Cancelled 상태는 마지막 상태로, Cancelled 상태로 전환되면 다시 다른 상태로 전환되지 않습니다.
Failing	COS에서 Failed로 진행되는 상태가 Failing 상태입니다.	Failed 상태로 전환됩니다.
Failed	작업 실패 시 Failed 상태가 됩니다. 작업 실패에 대한 자세한 정보는 실패 작업 추적을 참조하십시오.	Failed 상태는 마지막 상태로, Failed 상태로 전환되면 다시 다른 상태로 전환되지 않습니다.

실패 작업 추적

일괄 프로세스 작업 중 객체 리스트를 리졸브할 수 없는 등의 문제가 발생할 경우 일괄 프로세스 작업이 실패되고, COS에서 해당 에러 코드 및 오류 원인을 반환합니다. COS는 작업 실패 원인을 저장하며, DescribeJob API를 통해 작업 실패에 대한 자세한 정보를 확인할 수 있습니다. 또한 작업 보고서를 통해서도 관련 작업의 실패 사유 및 기타 관련 정보를 확인할 수 있습니다.

COS는 모든 일괄 프로세스 작업에 대해 작업 실패 임계값을 제공하여 생성한 작업에 대량의 실패 작업이 발생하지 않도록 방지합니다. 작업에 1000개 이상의 작업이 존재하는 경우 COS는 작업 실패율을 모니터링합니다. 언제든 작업 실패율(현재 진행 중인 작업 중 실패 작업에서 이미 진행된 모든 작업 수를 뺀 수)이 임계값인 50%를 초과하는 경우, COS는 작업을 중지하고 실패 상태를 반환합니다. 작업 실패율이 임계값을 초과한 이유를 필터링하여 확인할 수 있으며, 객체 리스트에 존재하지 않는 객체 정보가 대량으로 포함되어 있는 경우 오류 수정 후 다시 작업을 생성합니다.

주의 :
COS 일괄 프로세스 작업은 비동기화 방식으로 실행되며, 객체 작업이 반드시 리스트 상의 객체 순서로 실행되지 않습니다. 따라서 객체 리스트 상의 객체 순서에 따라 어떤 객체까지 작업되었다고 판단할 수 없으며, 이에 따라 작업의 성공 또는 실패 상황을 판단합니다. 작업 보고서에서 작업의 성공 또는 실패 정보를 확인할 수 있습니다.

작업 보고서

작업 생성 시 작업 보고서 출력 여부를 설정할 수 있습니다. 작업 보고서 출력을 설정한 경우, COS는 작업 성공, 실패 또는 취소 시 보고서를 출력하며, 해당 보고서로 작업과 관련된 모든 성공 또는 실패 작업 상황을 확인할 수 있습니다.

작업 보고서에는 작업의 지정 작업에 대한 설정 매개변수, 실행 상태 등의 정보가 포함되어 있으며, 이외에도 처리된 객체의 객체 이름, 버전 ID, 작업 상태 코드, 오류 설명 등의 내용이 포함되어 있습니다.

일괄 작업 개요

최종 업데이트 날짜: : 2023-04-27 16:01:33

Cloud Object Storage(COS)의 일괄 작업 기능을 사용하면 버킷 내에서 지정된 객체 목록에 대한 작업을 지정할 수 있습니다. 이를 위해 두 가지 옵션이 있습니다. 인벤토리 기능을 사용하여 객체 인벤토리를 생성하거나 인벤토리와 같은 형식으로 원하는 객체를 CSV 파일에 나열합니다. 이렇게 하면 COS에서 지정된 대로 객체가 일괄적으로 작업됩니다.

자세한 정보는 인벤토리 개요를 참고하십시오.

현재 COS 일괄 작업 기능은 다음과 같은 작업만 지원합니다.

- 객체 일괄 복사
- 보관 객체 일괄 복구

일괄 작업 기능은 COS 콘솔에서 사용할 수 있습니다. 자세한 내용은 일괄 작업을 참고하십시오.

원리

일괄 작업을 수행하려면 먼저 객체 목록에 대해 지정된 작업을 수행하기 위해 필요한 모든 정보를 포함하는 일괄 작업을 생성해야 합니다. 객체 목록으로 인벤토리를 사용할 수 있습니다.

인벤토리 파일을 제공하고 생성한 일괄 작업을 시작한 후, 일괄 작업 기능은 인벤토리의 객체에 지정된 작업을 순차적으로 수행합니다. 작업 실행 중에는 COS 콘솔에서 실행 상태를 모니터링하거나 작업이 완료된 후 작업 보고서를 출력할 수 있습니다. 작업 보고서는 작업 내 각 작업에 대한 자세한 정보를 제공합니다.

> 주의 :
> 일괄 작업 기능은 현재 버킷의 객체에만 적용됩니다. 다른 버킷의 객체에 대해 일괄 작업을 수행하려면 해당 버킷에 대해 일괄 작업 기능을 활성화해야 합니다.

객체 인벤토리

객체 인벤토리는 작업 대상인 모든 객체의 목록입니다. 일괄 작업을 생성하려면 먼저 COS에게 작업을 수행해야 할 객체를 알려주기 위해 객체 인벤토리를 제공해야 합니다. 객체 인벤토리 파일을 버킷에 넣고 파일 이름, ETag 및 VersionID(해당되는 경우)와 같은 정보를 제공해야 합니다. 객체 인벤토리는 다음 두 가지 방법으로 생성할 수 있습니다:

- **COS 인벤토리 기능**: 이 기능은 객체 인벤토리를 CSV 형식으로 출력합니다. 자세한 정보는 인벤토리 개요를 참고하십시오. 객체 인벤토리에 버전 ID 정보가 포함된 경우, COS는 해당 버전 ID를 가진 객체들에 대해 일괄 작업을 수행합니다.
- **CSV 파일 구성**: CSV 파일의 모든 행에는 일괄 작업을 위한 버킷 이름과 객체의 이름 및 버전 ID(버킷에 대해 버전 관리가 활성화된 경우)가 포함되어야 합니다. 버전 관리가 활성화된 적이 없는 경우 버전 ID 정보를 건너뛸 수 있습니다. CSV 파일은 다음과 같이 구성할 수 있습니다.

```
examplebucket-appid, exampleobject, PZ9ibn9D5lP6p298B7S9_ceqx1n5EJ0p
examplebucket-appid, exampleobject, jbo9_jhdPEyB4RrmOxWS0kU0EoNrU_oI
```

주의 :

- 만약 버전 관리가 버킷에 활성화되었거나 활성화되었던 적이 있으며, 지정된 버전의 객체에 대해 일괄 작업을 수행하려면, 객체 인벤토리에서 객체 버전 ID 정보를 제공해야 합니다.
- 만약 버전 관리가 버킷에 활성화되었거나 활성화되었던 적이 있지만 인벤토리에서 버전 ID를 지정하지 않은 경우, COS는 기본적으로 객체의 최신 버전에서 작업을 수행합니다.
- 작업을 생성하기 전에 객체 이름이 같은 객체를 업로드했다면, COS는 객체 인벤토리가 생성될 때의 버전이 아닌 최신 버전의 객체에서 기본적으로 작업을 수행합니다. 이 문제를 피하기 위해서는 버전 관리를 활성화하고 객체 인벤토리에서 버전 ID를 지정할 수 있습니다.
- 객체 인벤토리는 버킷의 모든 객체를 포함할 수 있습니다. 그러나 많은 수의 객체에 대해 작업을 수행하는 데에는 시간이 더 오래 걸릴 수 있다는 점을 유의해야 합니다.

일괄 작업

이 섹션에서는 일괄 작업을 생성하는 방법과 생성 후 시스템이 어떻게 응답하는지 설명합니다.

일괄 작업 생성 시에는 다음 정보를 제공해야 합니다.

유형	설명
작업	객체에 대해 수행할 작업을 지정해야 합니다. 각 작업에 대해 해당 매개변수를 구성할 수 있으며, COS는 인벤토리의 객체에 대해 구성된대로 순차적으로 작업을 수행합니다.
객체 인벤토리	객체 인벤토리는 수행할 모든 객체의 목록입니다. 인벤토리 기능을 사용하여 객체 인벤토리를 생성할 수 있습니다. 자세한 내용은 인벤토리 개요를 참고하십시오. 인벤토리와 동일한 형식으로 CSV 파일에서 필요한 객체를 나열할 수도 있습니다.

유형	설명
우선순위	우선순위를 설정하여 현재 일괄 작업의 우선 순위를 다른 작업보다 높게 설정할 수 있습니다. 작업 우선순위는 직접 작업이 완료되는 순서를 결정하지는 않습니다. 여러 작업의 순서를 제어하려면 직접 작업 실행 상태를 확인하고 현재 작업이 완료된 후 다음 작업을 시작해야 합니다.
규칙 권한	일괄 작업을 생성한 후 작업을 수행할 적절한 IAM 권한이 계정에 있는지 확인해야 합니다. 예를 들어 `PUT Object-copy` 를 실행하는 일괄 작업을 생성한 경우 소스 버킷에서 `Get Object` 권한과 대상 버킷에서 `PUT Object` 권한을 확인해야 합니다. 또한 모든 일괄 작업에 대해 객체 인벤토리를 읽고 작업 보고서에 쓰는 권한이 있어야 합니다. 권한 구성에 대한 자세한 내용은 권한 설정 및 버킷 액세스 정책을 참고하십시오.
작업 보고서	일괄 작업이 완료된 후 작업 보고서를 출력하려면 작업 생성 시 해당 매개변수를 입력하여 시스템이 작업 보고서를 지정된 대상 버킷에 올바르게 출력할 수 있도록 설정해야 합니다. 필요한 정보는 작업 보고서를 저장할 버킷, 작업 보고서 형식 및 모든 작업 정보 포함 여부입니다. 작업 보고서의 파일 접두사는 선택 사항입니다.
작업 설명 (옵션)	생성된 일괄 작업에 대해 256바이트 작업 설명을 입력하여 추적할 수 있습니다. 작업 설명은 COS 콘솔에서 표시되며 작업을 정렬하거나 필터링하는 데 사용할 수 있습니다. 유사한 작업(예: 매주 로그 데이터를 동기화하고 복사하는 것)에 동일한 작업 설명을 입력하여 중앙 집중식으로 관리할 수 있습니다.

일괄 프로세스 작업
객체 일괄 복사

최종 업데이트 날짜: : 2021-03-11 10:19:09

객체 일괄 복사 작업은 리스트 상의 객체를 복사하는데 사용됩니다. 지정 객체를 원본 버킷에서 타깃 버킷으로 일괄 복사할 수 있습니다. 타깃 버킷과 원본 버킷의 리전의 동일 여부에 관계 없이 실행 가능합니다. 객체 일괄 복사 작업은 `PUT Object-copy` 의 관련 매개변수를 사용자 지정 설정할 수 있으며, 해당 설정 정보는 복사본의 메타데이터 정보 또는 스토리지 유형 등의 정보에 영향을 미칩니다. PUT Object-copy 관련 자세한 내용은 PUT Object-copy를 참조하십시오.

관련 제한

- 모든 처리 대기 상태인 객체가 동일한 버킷에 있어야 합니다.
- 일괄 프로세스 작업별로 타깃 버킷은 한 개만 설정할 수 있습니다.
- 원본 버킷에 객체 읽기 권한과 타깃 버킷 객체 쓰기 권한을 부여해야 합니다.
- 처리 대기 중인 객체는 5GB를 초과할 수 없습니다.
- ETags 조회 및 사용자 지정 키를 사용한 서버 암호화를 지원하지 않습니다.
- 타깃 버킷에 버전 제어가 활성화되어 있지 않고 동일한 이름의 객체 파일이 존재하는 경우, COS는 객체 파일을 덮어씁니다.
- 리스트 상의 객체에 여러 버전이 존재하는 경우 하나의 버전만 복사하며, 버전을 지정하지 않은 경우 기본적으로 최신 버전만 복사합니다.

아카이브된 객체 일괄 복구

최종 업데이트 날짜: : 2022-11-14 14:28:36

보관된 객체를 일괄 복구하면 인벤토리에 보관된 객체를 복구하는 데 사용할 수 있습니다. 이 작업은 POST Object restore에 대한 구성 매개변수를 지원합니다. 구성 정보는 복사본의 복구 시간과 만료 시간에 영향을 미칩니다. 자세한 내용은 POST Object restore를 참고하십시오.

복구는 ARCHIVE 및 DEEP ARCHIVE 스토리지 클래스에 대해 지원됩니다. 이 두 가지 스토리지 클래스에 대한 자세한 내용은 스토리지 유형 개요를 참고하십시오.

보관된 객체 일괄 복구 작업을 생성할 경우 다음 두 가지 매개변수를 지정해야 합니다.

- 복구 모드: 표준 검색 또는 대량 검색. 자세한 내용은 아카이브된 객체 복구를 참고하십시오.
- 복사본 유효기간: 아카이브 유형의 객체는 복구 후 임시 복사본이 생성되며, 해당 복사본은 지정된 시간 후 자동으로 만료되어 삭제됩니다. 복사본 유효기간에 대한 자세한 내용은 아카이브된 객체 복구를 참고하십시오.

주의 사항

1. 일괄 복구 작업에 이미 복구된 객체의 복사본이 포함된 경우 작업이 시작될 때 이러한 복사본의 유효 기간이 업데이트되어 작업의 모든 복사본이 동일한 유효 기간을 갖도록 합니다.
2. 일괄 복구 작업은 객체 복구 요청만 시작합니다. 모든 요청이 전송된 후 일괄 작업 페이지에 작업이 완료된 것으로 표시됩니다. COS(Cloud Object Storage)는 객체가 복구되었을 때 사용자에게 알리지 않지만 알림을 수신하도록 이벤트 알림을 구성할 수 있습니다. 자세한 내용은 이벤트 알림을 참고하십시오.

글로벌 가속
글로벌 가속 개요

최종 업데이트 날짜: : 2022-11-15 16:23:44

Tencent Cloud Cloud Object Storage(COS)의 글로벌 가속 기능은, Tencent의 전체 트래픽을 스케쥴링하는 CLB(Cloud Load Balancer) 시스템을 통해, 사용자의 요청을 스마트 라우팅 및 리졸브하여, 최적의 네트워크 액세스 링크를 선택하고, 요청에 따른 근접 액세스를 구현합니다. 전 세계에 분포한 클라우드 데이터 센터가 전 세계 각지의 사용자의 신속한 버킷 액세스를 구현하여, 액세스 성공률이 높을 뿐만 아니라, 안정적인 비즈니스 환경을 제공하고, 비즈니스 경험을 향상시킵니다. 또한 COS의 글로벌 가속 기능으로 더욱 신속하게 데이터를 업로드 및 다운로드할 수 있습니다.

주의 :

- 현재 글로벌 가속 기능은 모든 퍼블릭 클라우드 리전에서 공급 및 지원됩니다.
- 글로벌 가속 기능을 사용하면, 정보 요청이 Tencent Cloud 내부 네트워크 전용 회선을 통해 가속 전송되기 때문에 요금이 부과됩니다. 자세한 요금 정보는 제품 가격을 참고하십시오.

사용 방법

COS 콘솔과 API 등 방식으로 글로벌 가속 기능을 활성화할 수 있습니다.

COS 콘솔 사용

COS 콘솔에서 버킷에 필요한 글로벌 가속 기능을 활성화할 수 있습니다. 글로벌 가속 활성화 콘솔 문서를 참고하십시오.

REST API 사용

다음의 API를 통해 글로벌 가속 기능을 직접 활성화할 수 있습니다.

- PUT Bucket Accelerate
- GET Bucket Accelerate

액세스 도메인

글로벌 가속 기능을 활성화한 뒤, 두 가지 도메인으로 COS 내 파일에 액세스할 수 있습니다.

- **버킷 기본 도메인: **형식은 `<bucketname-appid>.cos.<region>.myqcloud.com` 이며, 자세한 내용은 리전 및 액세스 도메인을 참고하십시오.
- **글로벌 가속 도메인: **형식은 `<bucketname-appid>.cos.accelerate.myqcloud.com` 입니다.

예를 들어, 광저우 리전의 버킷 `examplebucket-1250000000` 에 글로벌 가속 기능을 활성화한 경우, 베이징에서 해당 버킷으로 파일 `exampleObject.txt` 를 업로드할 때 선택할 수 있는 두 가지 업로드 방식이 있습니다.

- **글로벌 가속 도메인으로 액세스**: 업로드할 때 도메인을 `exampleBucket-1250000000.cos.accelerate.myqcloud.com` 으로 지정하면 해당 도메인을 통해 객체를 업로드할 경우, COS 서비스는 네트워크 상태에 따라 스마트 리졸브 및 근접 액세스를 실행합니다. 예를 들어, 베이징의 액세스 레이어로 요청을 전달하면, 내부 네트워크의 전용 회선을 거쳐 광저우 스토리지 레이어로 전송되는 과정이 가속으로 이뤄집니다.
- **버킷 기본 도메인으로 액세스**: 업로드 시 도메인을 `examplebucket-125000000.cos.ap-guangzhou.myqcloud.com` 으로 지정합니다. 해당 도메인을 통해 객체를 업로드할 때, 요청이 광저우의 액세스 레이어로 바로 전달된 뒤, 다시 광저우 스토리지 레이어로 전송됩니다. 이때 외부 네트워크의 링크가 길기 때문에 전송이 불안정할 수 있습니다.

> 주의 :
> 글로벌 가속 기능은 별도의 요금이 부과되므로 실제 비즈니스 상황을 잘 고려하여 선택하시기 바랍니다.
>
> 1. 평소 읽기 작업보다 쓰기 작업을 많이 수행하고, 원격지에서 클라우드 데이터 센터로 데이터를 업로드(예: PUT Object, POST Object, Multipart Upload 작업)하는 경우, 글로벌 가속 도메인 사용을 권장합니다.
> 2. 평소 쓰기 작업보다 읽기 작업을 많이 수행하고, 주로 파일을 다운로드(예: GET Objcet 작업)하는 경우, 비용 최적화를 위해, CDN 가속 솔루션을 권장합니다.
> 3. 평소 설정이나 파일 인덱스 작업이 많다면, 버킷의 기본 도메인 사용을 권장합니다.
> 4. 동일 리전의 내부 네트워크 환경에서 동일 리전 내 버킷을 액세스하거나 전용 회선으로 액세스할 경우, 버킷의 기본 도메인 사용을 권장합니다.

주의 사항

다음은 글로벌 가속 도메인 사용 시 주의사항입니다.

- 글로벌 가속 도메인을 활성화하고 약 15분 후에 적용되므로 잠시 기다려주십시오.
- 글로벌 가속 도메인을 활성화한 뒤 단일 버킷이 가속 도메인으로 액세스하는 최대 대역폭은 전체 네트워크 사용량에 따라 분배됩니다.
- 글로벌 가속 도메인을 활성화하면 가속 도메인에 의한 요청만 가속 효과가 적용되며, 버킷의 기본 도메인은 여전히 정상적으로 사용할 수 있습니다.

- 가속 도메인을 사용하는 경우, 요청 링크가 가속 링크일 때만 가속 요금이 부과됩니다. 예를 들어, 가속 도메인으로 베이징에서 베이징의 버킷으로 데이터를 업로드하면 링크에 대해 가속 기능이 작동하지 않아 요청 관련 요금이 발생하지 않습니다.
- 가속 도메인을 사용하는 경우, HTTP/HTTPS 전송 프로토콜을 지정하여 내부 네트워크의 전용 회선으로 정보 요청을 전송하면 COS에서 데이터 전송 보안을 위한 HTTPS 프로토콜의 채택 여부를 판단합니다.

과금 예시

글로벌 가속 도메인으로 데이터를 업로드하거나 버킷에 액세스하면 별도의 가속 요금이 부과되며, 가속 요금은 **일별** 결산됩니다. 관련 과금 항목 및 가격 정보는 과금 개요와 제품 가격을 참고하십시오. 다음은 가속 도메인과 버킷 기본 도메인을 사용했을 때의 요금 비교 내역입니다.

비즈니스 시나리오 1

주로 비디오 파일을 COS에 업로드하는 시나리오로, 전송 성공률에 대한 요구도가 높습니다. 광저우 리전에 설정된 버킷에 매일 신장과 싱가포르의 사무실로부터 각각 1GB의 비디오 데이터가 업로드됩니다. 이 경우, 30일 간의 요금을 계산하면 다음과 같습니다.

- **가속 도메인**으로 업로드 시 발생하는 업로드 요금: 30 x 1GB x (0.07USD/GB + 0.18USD/GB) = 7.5USD
- **버킷 기본 도메인**으로 업로드 시 발생하는 업로드 요금: 30 x 1GB x (0USD/GB) = 0USD

> 설명 :
> 중국 내의 업로드 가속 요금 단가는 0.07USD/GB이며, 중국 외 지역의 업로드 가속 요금은 0.18USD/GB입니다. 버킷 기본 도메인으로 파일 업로드 시 업로드 트래픽 요금이 발생하지 않습니다.

비즈니스 시나리오 2

해외 파일 다운로드 가속 시나리오로, 전송 성공률에 대한 요구도가 높습니다. 버킷은 광저우 리전에 설정되어있으며, 매일 싱가포르 사무실로부터 1GB의 비디오 데이터를 다운로드합니다. 이 경우, 30일 간의 요금을 계산하면 다음과 같습니다.

- 가속 도메인으로 다운로드 시 발생하는 다운로드 가속 트래픽 요금 :30 x 1GB x 0.18USD/GB = 5.4USD
- 가속 도메인으로 다운로드 시 발생하는 공인 네트워크 다운스트림 트래픽 요금:30 x 1GB x 0.1USD/GB = 3USD

총 다운로드 트래픽 요금은 5.4 + 3 = 8.4USD입니다.

> 설명 :

해외 다운로드 가속 요금 단가는 0.18USD/GB입니다. 글로벌 가속 도메인으로 파일 다운로드 시 **외부 네트워크 다운스트림 트래픽 비용**과 **글로벌 가속 다운스트림 트래픽 비용**이 청구됩니다.

사설망 글로벌 가속

최종 업데이트 날짜: : 2023-03-14 17:05:32

Tencent Cloud의 글로벌 트래픽 스케쥴링 기능과 COS의 사설망 글로벌 가속화 경로를 활용하여 사용자 요청을 지능적으로 해결하여 최적의 네트워크 액세스 링크를 선택하고 개별 리전 간의 리소스에 빠르게 액세스할 수 있도록 지원합니다. 이를 통해 Tencent Cloud의 사설망 백본 라인의 안정적인 전송 품질을 충분히 활용하여 리전 간 데이터 전송의 안정성과 성능을 향상시킬 수 있습니다.

많은 비즈니스는 사설망을 통해 리전 간 데이터를 가져와야 하는 필요성이 있습니다. 예를 들어, 컨테이너 이미지 배포 시나리오에서는 이미지 리포지토리는 보통 한 리전에 중앙 집중식으로 저장되며, 컨테이너 클러스터는 비즈니스 요구에 따라 다른 리전에 배포될 수 있습니다. 애플리케이션 배포 중에는 컨테이너 클러스터가 다른 리전에서 이미지를 가져와야 하므로 많은 리전 간 요청이 발생합니다. 이러한 시나리오에서 사설망 글로벌 가속화 엔드포인트를 사용하여 안정적인 사설망 리전 간 직접 연결 라인을 제공하여 이미지 배포 안정성을 향상시킬 수 있습니다.

> 주의 :
> 이 기능을 사용하면 요청 데이터 전송이 Tencent Cloud 사설망의 직접 연결 회선을 통해 가속화되므로 요금이 발생합니다. 자세한 가격 정보는 가격 | Cloud Object Storage를 참고하십시오.

사용 방법

COS 콘솔과 API 등 방식으로 글로벌 가속 기능을 활성화할 수 있습니다.

COS 콘솔 사용

COS 콘솔에서 버킷에 필요한 글로벌 가속 기능을 활성화할 수 있습니다. 글로벌 가속 활성화 콘솔 문서를 참고하십시오.

REST API 사용

다음의 API를 통해 글로벌 가속 기능을 직접 활성화할 수 있습니다.

- PUT Bucket Accelerate
- GET Bucket Accelerate

도메인 이름 액세스

글로벌 가속 기능을 활성화한 후 `<bucketname-appid>.cos-internal.accelerate.tencentcos.cn` 형식의 사설망 글로벌 가속 엔드포인트를 통해 사설망을 통해 다른 리전의 COS 파일에 액세스할 수 있습니다.

예를 들어, 어떤 비즈니스 애플리케이션 이미지가 광저우 리전에 있는 `examplebucket-1250000000` 버킷에 저장되어 있으며, 해당 애플리케이션의 컨테이너 클러스터가 광저우, 베이징, 상하이 리전에 배포되어 있다고 가정합니다. 컨테이너 이미지 배포를 가속화하기 위해 비즈니스 팀은 버킷 `examplebucket-1250000000` 에 대한 글로벌 가속화를 활성화하여, 베이징, 상하이 및 광저우 리전의 컨테이너 클러스터가 사설망 글로벌 가속화 엔드포인트 `examplebucket-1250000000.cos-internal.accelerate.tencentcos.cn` 를 통해 사설망을 통해 이미지 패키지를 가져올 수 있게 되었습니다.

- 광저우 리전의 컨테이너 클러스터가 엔드포인트를 통해 파일을 가져오면 COS는 광저우 리전의 사설망 액세스 레이어에 대한 요청을 지능적으로 해결하여 동일한 리전의 스토리지 클러스터에서 직접 파일을 풀링합니다.
- 베이징/상하이 리전의 컨테이너 클러스터가 엔드포인트를 통해 파일을 가져올 때, COS는 베이징/상하이 리전의 사설망 액세스 레이어에 대한 요청을 지능적으로 리졸브하고 사설망 액세스 레이어 장치를 사용하여 광저우 리전의 스토리지 클러스터에서 파일을 리전 간 백본 네트워크 직접 연결 회선을 통해 풀링합니다.

주의 :

- 글로벌 가속 기능은 별도의 요금이 부과되므로 실제 비즈니스 상황을 잘 고려하여 선택하시기 바랍니다.
- 사설망 글로벌 가속 엔드포인트는 Tencent Cloud 사설망 환경에서만 사용할 수 있습니다. 요청 소스가 Tencent Cloud 사설망 환경에 있지 않으면 연결할 수 없습니다. 이 경우 기본 엔드포인트 또는 기본 글로벌 가속 엔드포인트 사용을 고려할 수 있습니다.
- CVM, TKE 또는 SCF와 같은 다른 Tencent Cloud 제품에서 사설망 글로벌 가속 엔드포인트를 사용하는 경우 해당 제품은 COS를 사용할 수 있는 리전에 있어야 하며, 그렇지 않으면 사용할 수 없습니다. COS를 사용할 수 있는 리전에 대한 자세한 내용은 리전 및 액세스 도메인을 참고하십시오.

주의 사항

다음은 글로벌 가속 도메인 사용 시 주의사항입니다.

- 글로벌 가속 도메인을 활성화하고 약 15분 후에 적용되므로 잠시 기다려주십시오.
- 글로벌 가속 도메인을 활성화한 뒤 단일 버킷이 가속 도메인으로 액세스하는 최대 대역폭은 전체 네트워크 사용량에 따라 분배됩니다.
- 글로벌 가속 도메인을 활성화하면 가속 도메인에 의한 요청만 가속 효과가 적용되며, 버킷의 기본 도메인은 여전히 정상적으로 사용할 수 있습니다.
- 가속 도메인을 사용하는 경우, 요청 링크가 가속 링크일 때만 가속 요금이 부과됩니다. 예를 들어, 가속 도메인으로 베이징에서 베이징의 버킷으로 데이터를 업로드하면 링크에 대해 가속 기능이 작동하지 않아 요청 관련 요금이 발

생하지 않습니다.

- 가속 도메인을 사용하는 경우, HTTP/HTTPS 전송 프로토콜을 지정하여 내부 네트워크의 전용 회선으로 정보 요청을 전송하면 COS에서 데이터 전송 보안을 위한 HTTPS 프로토콜의 채택 여부를 판단합니다.

데이터 처리
이미지 처리 개요

최종 업데이트 날짜: : 2023-03-14 17:05:32

소개

이미지 처리는 Cloud Infinite(CI)에서 제공하는 이미지 처리 기능의 세트입니다. 이미지 자르기, 포맷 변환, 스케일링, 워터마크 등의 기본적인 처리 기능과 Guetzli 압축, AVIF 트랜스코딩 및 압축과 같은 이미지 다운사이징 기능, 저작권 보호를 위한 블라인드 워터마크, 이미지 향상, 태그, 평가, 복구 및 이미지 매팅과 같은 AI 기반 인식 및 분석 기능 등을 지원하여 다양한 비즈니스 시나리오에서 이미지 처리 요구 사항을 충족합니다.

주의 :

- 이미지 처리 기능은 퍼블릭 클라우드 리전만 지원합니다.
- 이미지 처리 기능은 CI에서 과금하는 과금 항목입니다. 자세한 과금 설명은 Billing Overview를 참고하십시오.
- 현재 다중 AZ 버킷에 대해 이미지 처리가 지원되지 않습니다.

서비스	기능	설명
기본 이미지 처리 서비스	크기 조절	동일 비율 조정, 타깃의 가로:세로 비율 설정 등 다양한 방법
	자르기	일반 자르기, 크기 조정 자르기, 내접원 자르기, 스마트 얼굴 자르기
	회전	자동 회전, 일반 회전
	형식 변환	포맷 변환, GIF 포맷 최적화, 점진적 표시
	품질 변경	JPG 및 WEBP 이미지 품질 변경
	가우시안 블러	이미지 가우시안 블러 처리
	샤프닝	이미지 샤프닝 처리

	워터마크 추가	이미지 워터마크, 텍스트 워터마크
	이미지 정보 가져오기	기본 정보, EXIF 정보, 메인 컬러
	메타 정보 삭제	EXIF 정보 포함
	빠른 섬네일 템플릿	이미지의 빠른 포맷 변환, 축소, 자르기 등 기능 구현 및 썸네일 생성
	스타일 설정	이미지 스타일 설정으로 다양한 니즈의 이미지를 편리하게 관리
AI 기반 이미지 인식	이미지 복구	이미지에서 LOGO, 객체 및 워터마크와 같은 콘텐츠를 제거하고 그 부분을 배경으로 지능적으로 채워 이미지의 특정 영역을 효과적으로 복구합니다.
	이미지 매트	이미지에서 신체 부위를 지능적으로 인식하고 나머지 부분은 투명하게 만듭니다.
	Logo 인식	이미지에서 브랜드 Logo를 인식하고 이미지에서 Logo 문자 및 위치와 같은 정보를 반환합니다.
	QR 코드 인식	이미지에서 QR 코드를 인식하고 해당 위치와 내용을 반환합니다. 인식된 QR 코드를 픽셀화할 수 있습니다.
	이미지 태그	이미지에서 장면, 물체, 동물(고양이, 개, 새 포함), 음식(과일 및 채소 포함)과 같은 정보를 지능적으로 인식하고 해당 태그를 추가합니다. 수십 개의 카테고리에서 수천 개의 태그가 지원됩니다.
	이미지 품질 평가	다양한 차원에서 시각적 이미지 품질을 평가하고 객관적인 해상도 점수와 주관적인 미적 점수를 출력합니다.
	얼굴 인식	주어진 얼굴 이미지에서 얼굴 위치, 얼굴 특징, 얼굴 품질 정보를 감지하고 뷰티 필터, 인물 사진 키잉, 나이 변경, 성별 전환 등 다양한 특수 효과를 지원합니다.
	페이스 ID	ID 카드 인식 및 얼굴 생체 인식과 같은 기능을 제공합니다.
	차량 인식	이미지에서 차량을 감지하고 차량 브랜드, 색상, 위치 및 번호판 번호와 같은 정보를 인식합니다.
	텍스트 인식	이미지의 단어를 지능적으로 인식하고 편집 가능한 텍스트로 변환합니다.

	이미지로 검색	버킷에 이미지 라이브러리를 생성하고 지정된 이미지 라이브러리에서 동일하고 유사한 이미지를 빠르게 검색합니다.
기타	비정상 이미지 감지	TS 비디오 스트림이 포함된 이미지와 같이 비정상적이고 의심스러운 정보가 포함된 이미지를 감지합니다.

사용 방법

COS 콘솔 사용

COS 콘솔에서 기본 이미지 처리 작업을 수행할 수 있습니다. 자세한 내용은 기본 이미지 처리를 참고하십시오.

REST API 사용

COS에서 제공하는 API를 이용하여 기본적인 이미지 처리나 AI 기반 인식을 수행할 수 있습니다. 자세한 내용은 Data Processing APIs를 참고하십시오.

제한 설명

- 지원 형식: JPG, BMP, GIF, PNG, WEBP 이미지 처리와 HEIF 이미지 디코딩 및 처리가 가능합니다.
- 용량 제한: 원본 이미지의 크기는 32MB, 가로 x 세로는 30000픽셀, 총 픽셀은 2.5억 픽셀을 초과할 수 없으며, 처리 완료된 이미지의 가로 x 세로는 9999픽셀을 초과해 설정할 수 없습니다. 애니메이션 이미지의 경우 원본 이미지의 가로 x 세로 x 프레임 수는 2.5억 픽셀을 초과할 수 없습니다.
- 프레임 수(애니메이션 이미지의 경우): gif의 경우 프레임 수는 300개를 초과할 수 없습니다.

이미지 압축 개요

최종 업데이트 날짜: : 2023-04-27 16:01:33

개요

이미지 압축이란 이미지의 저장 및 트래픽 비용을 줄이고 액세스 속도를 높이기 위해 품질을 변경하지 않고 이미지의 크기를 최대한 줄이는 것을 말합니다.

Cloud Object Storage(COS)는 비즈니스 시나리오에 따라 Cloud Infinite(CI) 기반의 다양한 이미지 압축 방식을 출시했습니다. 현재 지원되는 압축 방법은 다음과 같습니다.

- **AVIF 압축:** 2020년 2월 av1 기반으로 Netflix에서 출시한 새로운 이미지 형식인 avif 형식으로 이미지를 변환하며 현재 Chrome 및 Firefox와 같은 브라우저에서 지원됩니다.
- **WebP 압축:** 압축 측면에서 jpg보다 우수한 webp 형식으로 이미지를 변환합니다. webp 이미지는 동일한 품질의 jpg 이미지보다 25% 이상 작습니다. 이 형식은 다중 터미널 사용 사례에 적합합니다.
- **HEIF 압축:** 이미지를 압축률이 매우 높은 heif 형식으로 변환합니다. webp 이미지는 동일한 품질의 jpg 이미지보다 80% 이상 작습니다. iOS는 heif를 사진의 기본 형식으로 채택하고 Android P는 기본적으로 heif를 지원합니다.
- **TPG 압축:** 이미지를 tpg 형식으로 변환합니다. 이 형식은 Tencent에서 출시한 독점 이미지 형식이며 애니메이션 이미지를 지원합니다. 현재 QQ Browser, Qzone 및 기타 Tencent 제품은 기본적으로 tpg를 지원합니다. tpg 이미지는 동일한 품질의 gif 또는 png 이미지보다 각각 90% 이상 또는 50% 이상 작습니다.
- **스마트 이미지 압축**: 이미지의 주관적인 품질을 지능적으로 결정하고 자동으로 조정합니다. 원본 형식을 변경하지 않고 이미지 크기를 크게 줄여 원본 이미지에 최대한 가까운 시각적 효과를 제공합니다.

설명 :

- 이미지 압축은 CI에서 과금하는 유료 서비스입니다. 자세한 가격은 이미지 처리 요금을 참고하십시오.
- 현재 스마트 이미지 압축 서비스는 베이징과 상하이 리전에서만 사용할 수 있습니다.

사용 사례

이미지 압축 기능은 전자 상거래 및 미디어와 같은 다양한 사용 사례에서 PC 및 App과 같은 다양한 터미널에서 이미지 압축 요구를 충족합니다. 이는 전송 시간, 로딩 시간, 대역폭 및 트래픽 사용을 효과적으로 줄입니다.

다양한 압축 기능은 아래에 설명된 대로 기존 이미지 형식 및 브라우저 환경과의 호환성이 다릅니다.

기능	지원 형식	지원 브라우저 및 시스템	호환성	압축 효과	압축 속도
AVIF 압축	jpg, png, bmp, gif, heif, webp, tpg.	Firefox, Chrome 및 Android와 같은 일부 브라우저 및 시스템	비교적 강함	매우 강함	빠름
WebP 압축	jpg, png, bmp, gif, heif, tpg, avif.	Edge, Firefox, Chrome, Safari, Android, iOS 및 WeChat과 같은 브라우저 및 시스템의 95% 이상	강함	평균	빠름
HEIF 압축	jpg, png, bmp, webp, avif.	브라우저에서는 지원되지 않지만 iOS 11 이상 및 Android P에서 기본적으로 지원됨	약함	강함	빠름
TPG 압축	jpg, png, bmp, gif, heif, webp, avif.	특수 디코더로 QQ 브라우저와 같은 몇 가지 브라우저만 필요	약함	강함	빠름
스마트 이미지 압축	jpg 및 png(원본 이미지의 형식을 변경하지 않음).	모두	매우 강함	강함	빠름

설명：
CI는 TPG 및 AVIF 디코더를 통합하는 Windows SDK를 제공하므로 클라이언트에 통합하기만 하면 TPG 및 AVIF 이미지를 디코딩하고 미리보기 할 수 있습니다.

사용 방법

AVIF, HEIF 및 TPG 압축

이 세 가지는 고급 이미지 형식이며 사용하기 전에 먼저 이미지 고급 압축 기능을 활성화해야 합니다.

고급 압축 기능을 활성화한 후 이미지 형식 변환 매개변수를 원하는 압축 format으로 설정하여 사용할 수 있습니다. 구체적인 매개변수는 다음과 같습니다.

매개변수	설명
imageMogr2/format/avif	압축을 위해 원본 이미지를 avif 형식으로 변환합니다.

매개변수	설명
imageMogr2/format/heif	압축을 위해 원본 이미지를 heif 형식으로 변환합니다.
imageMogr2/format/tpg	압축을 위해 원본 이미지를 tpg 형식으로 변환합니다.

WebP 압축

기본 이미지 처리의 형식 변환 기능을 통해 WebP 압축 기능을 바로 사용할 수 있습니다. 구체적인 매개변수는 다음과 같습니다.

매개변수	설명
imageMogr2/format/webp	압축을 위해 원본 이미지를 webp 형식으로 변환합니다.

스마트 이미지 압축

스마트 이미지 압축 기능은 이미지에 액세스하는 방식을 변경하거나 압축 매개변수를 추가하지 않고 일치하는 형식으로 이미지를 자동으로 압축합니다.

콘솔을 사용하여 스마트 이미지 압축 기능 활성화해야 합니다. 기능이 활성화되면 이전과 동일한 방식으로 이미지에 액세스할 수 있으며 이미지가 자동으로 압축됩니다.

압축 예시

원본 png 이미지에 상기 모든 압축 작업을 수행합니다. 원본 이미지의 링크가 다음과 같다고 가정합니다.

https://examples-125xxxxx.cos.ap-shanghai.myqcloud.com/test.png

예시1: jpeg 형식으로 변환

최종 요청 URL:

```
https://examples-125xxxxx.cos.ap-shanghai.myqcloud.com/test.png?imageMogr2/format/jpeg
```

예시2: webp 형식으로 변환

최종 요청 URL:

```
https://examples-125xxxxx.cos.ap-shanghai.myqcloud.com/test.png?imageMogr2/format/webp
```

예시3: heif 형식으로 변환

최종 요청 URL:

```
https://examples-125xxxxx.cos.ap-shanghai.myqcloud.com/test.png?imageMogr2/format/heif
```

예시4: tpg 형식으로 변환

최종 요청 URL:

```
https://examples-125xxxxx.cos.ap-shanghai.myqcloud.com/test.png?imageMogr2/format/tpg
```

예시5: avif 형식으로 변환

최종 요청 URL:

```
https://examples-125xxxxx.cos.ap-shanghai.myqcloud.com/test.png?imageMogr2/format/avif
```

예시6: 스마트 이미지 압축 진행

최종 요청 URL:

```
https://examples-125xxxxx.cos.ap-shanghai.myqcloud.com/test.png
```

다음 표는 다양한 이미지 압축 옵션의 압축률을 비교한 것입니다(값은 참고용임).

형식	크기
png(원본 이미지)	465 KB
jpeg	114KB(75.5% 더 작음)
avif	32 KB(93.1% 더 작음)
webp	64 KB(86.2% 더 작음)
heif	54 KB(88.4% 더 작음)
tpg	56 KB(88.0% 더 작음)
스마트 이미지 압축	59 KB(87.3% 감소)

문서 미리보기 개요

최종 업데이트 날짜: : 2023-03-06 11:44:40

개요

문서 미리보기 기능은 Tencent Cloud CI(Cloud Infinite)를 기반으로 파일을 이미지, PDF 또는 HTML5 페이지로 트랜스 코딩할 수 있는 기능을 제공하여 문서 콘텐츠의 페이지 표시 문제를 해결하고 PC, App 등 여러 클라이언트의 문서의 **온라인 브라우징** 요구를 충족하며 온라인 교육, 기업 OA, 웹사이트 트랜스 코딩 등 비즈니스 시나리오에 적합합니다.

설명 :

- 현재 지원되는 입력 파일 형식은 다음 형식과 같습니다:
  - 데모 파일: pptx, ppt, pot, potx, pps, ppsx, dps, dpt, pptm, potm, ppsm.
  - 텍스트 파일: doc, dot, wps, wpt, docx, dotx, docm, dotm.
  - 테이블 파일: xls, xlt, et, ett, xlsx, xltx, csv, xlsb, xlsm, xltm, ets.
  - 기타 형식 파일: pdf, lrc, c, cpp, h, asm, s, java, asp, bat, bas, prg, cmd, rtf, txt, log, xml, htm, html.
- 현재 상기 파일 형식을 jpg, png, pdf, html 형식으로 트랜스 코딩할 수 있습니다.
- 입력 파일의 크기는 200MB로 제한되며, 입력 파일의 페이지 수는 5000페이지로 제한됩니다.

아키텍처

현재 문서 미리보기 기능은 동기화 트랜스 코딩과 비동기화 트랜스 코딩의 두 가지 방법을 제공합니다.

적용 시나리오

PC 및 App 등 여러 클라이언트의 문서를 온라인 브라우징 요구에 맞추어, 온라인 교육, 기업 OA 및 웹 사이트 트랜스 코딩과 같은 비즈니스 시나리오에 적합합니다.

사용 방법

COS 콘솔 사용

문서 미리보기 기능을 사용하기 전에 문서 미리보기 서비스를 활성화해야 하며, 자세한 내용은 문서 미리보기 콘솔 활성화 가이드 문서를 참고하십시오.

비동기화 트랜스 코딩

COS 콘솔을 사용하여 비동기화 문서 트랜스 코딩 미리보기 작업을 수행할 수 있으며, 자세한 내용은 파일 미리보기 작업 생성 콘솔 가이드 문서를 참고하십시오.

REST API 사용

동기화 트랜스 코딩

API를 사용하여 버킷에 있는 문서의 실시간 트랜스 코딩을 미리 볼 수 있습니다.

- HTML5가 아닌 미리보기를 사용합니다. 자세한 내용은 동기화 요청 인터페이스 API 문서를 참고하십시오.
- HTML5 미리보기를 사용합니다. 자세한 내용은 문서 전환 HTML 시작하기 API 문서를 참고하십시오.

블라인드 워터마크 개요

최종 업데이트 날짜: : 2022-07-04 11:01:27

개요

블라인드 워터마크 기능은 새로운 워터마크 모드인 Tencent Cloud CI(Cloud Infinite)를 기반으로 합니다. 이 기능을 통해 원본 이미지의 품질에 큰 영향을 주지 않으면서 원본 이미지 정보에 워터마크 이미지를 보이지 않는 형태로 추가할 수 있습니다. 이미지 도용 후, 도용이 의심되는 리소스에 대해 블라인드 워터마크 추출을 수행하여 이미지의 속성을 확인할 수 있습니다.

블라인드 워터마크 기능에는 세 가지 유형의 세미 블라인드, 퍼펙트 블라인드 및 텍스트 블라인드 워터마크가 있습니다.

워터마크 유형	특징	적용 시나리오
세미 블라인드 워터마크 (type1)	강력한 도난 방지 효과가 있지만 워터마크 추출 시 원본 이미지 필요	작은 이미지(640x640 이하)
퍼펙트 블라인드 워터마크 (type2)	추출이 용이하고 워터마크 추출 시 이미지 워터마크 자체만 필요	일괄 추가 및 확인
텍스트 블라인드 워터마크 (type3)	이미지에 텍스트 추가	단말 정보 추가

설명 :

- 블라인드 워터마크는 유료 서비스로 버킷의 구성 페이지에서 활성화 버튼을 통해 활성화해야 합니다.
- 블라인드 워터마크는 모든 퍼블릭 클라우드 리전에서 사용할 수 있습니다.

적용 시나리오

- **인증 및 책임**: 이미지에 세미 블라인드 워터마크를 추가하면 도난 시 이미지에 대한 소유권을 주장할 수 있으며, 해당 입력 이미지를 사용하여 블라인드 워터마크를 추출하여 이미지를 소유하고 있음을 증명할 수 있습니다.

- **중복 업로드 확인**: 경우에 따라 다른 사용자가 중복 이미지(예: 부동산 이미지, 자동차 이미지, 제품 이미지)를 업로드할 수 있습니다. 이 문제를 해결하기 위해 업로드 전에 퍼펙트 블라인드 워터마크 추출을 진행하여, 그 결과 이미지에서 블라인드 워터마크를 추출할 수 있으면 이전에 업로드된 이미지 입니다. 이 경우 사용자에게 중복 이미지

를 업로드하지 않도록 안내하는 등 해당 작업을 수행할 수 있습니다. 이미지에 블라인드 워터마크가 포함되어 있지 않으면 이미지에 추가하여 중복 업로드를 방지할 수 있습니다.

- **이미지 유출 방지**: 내부 이미지의 경우 텍스트 블라인드 워터마크를 사용하여 이미지에 요청자에 대한 정보를 추가할 수 있습니다. 이런 식으로 이미지가 유출되었을 때 블라인드 워터마크를 추출하여 유출자에 대한 정보를 얻을 수 있습니다.

관련 설명

- 현재 GIF와 같은 애니메이션 이미지에는 블라인드 워터마크를 추가할 수 없습니다.
- 이미지 워터마크의 너비와 높이는 모두 원본 이미지의 1/8을 초과해서는 안 됩니다.
- 블라인드 워터마크 효과를 보장하기 위해 검정색 배경의 흰색 워터마크를 선택해야 합니다.
- 이 서비스를 계정으로 처음 사용하는 경우 CI는 2개월 동안 유효한 6000회 무료 리소스 팩을 발급하며 리소스 팩을 초과한 사용량은 정상 과금됩니다. 자세한 내용은 CI 프리 티어를 참고하십시오.
- 텍스트 블라인드 워터마크는 현재 숫자[0 - 9]와 영문 대소문자[A - Z, a - z]를 지원합니다.
- 블라인드 워터마크는 자르기, 스머징, 색상 변경과 같은 다양한 도난 공격으로부터 이미지를 보호합니다. 도난 방지 효과는 원본 이미지의 크기와 공격 강도의 영향을 받습니다. 자세한 내용은 고객센터로 문의주시기 바랍니다.

사용 방법

COS 콘솔 사용

블라인드 워터마크는 COS 콘솔을 통해 활성화할 수 있으며, 자세한 사항은 블라인드 워터마크 콘솔 설정 가이드 문서를 참고하십시오.

REST API 사용

API를 사용하여 블라인드 워터마크를 추가하거나 추출할 수 있습니다. 자세한 내용은 Blind Watermarking API 문서를 참고하십시오.

미디어 처리 개요

최종 업데이트 날짜: : 2022-09-09 10:13:22

소개

미디어 처리는 Cloud Object Storage(COS)에서 CI(Cloud Infinite) 기반으로 제공하는 멀티미디어 파일 처리 서비스입니다. 오디오/비디오 트랜스 코딩, 비디오 프레임 캡처, 스마트 썸네일과 같은 Tencent Cloud의 최첨단 AI 기술로 강화된 다양한 기능을 제공합니다.

기능	설명
멀티미디어 트랜스 코딩	멀티미디어 파일 비트 스트림을 변환합니다. 코덱, 해상도 및 비트 레이트와 같은 원본 비트 스트림의 매개변수를 변경하여 다양한 장치 및 네트워크 조건에 적응합니다.
TESHD 트랜스 코딩	비디오를 더 작고 선명하게 만드는 트랜스 코딩 기능을 제공합니다. 이미지 리마스터링 및 향상, 적응형 콘텐츠 매개변수 선택, V265 인코더와 같은 완전한 비디오 처리 솔루션 세트를 통합하여 낮은 네트워크 리소스 사용량을 보장하면서 더 나은 시각적 경험을 제공합니다.
전문 미디어 형식 트랜스 코딩	XAVC 및 ProRes와 같은 특수 형식을 트랜스 코딩할 수 있습니다.
비디오 몽타주	비디오 콘텐츠, 동작, 자세 및 장면을 인식 및 집계하고 전문적으로 빠르게 클리핑하여 비디오에서 하이라이트를 정확하게 추출합니다.
비디오 향상	디테일 향상, 색상 향상, SDR to HDR을 포함한 일련의 기능을 통해 비디오 이미지 품질을 최적화하고 시각 효과를 향상시킵니다.
슈퍼 해상도	일련의 저해상도 비디오 이미지를 통해 고해상도 비디오 이미지를 생성하기 위해 비디오의 내용과 윤곽을 인식하여 비디오의 세부 사항 및 국부적 특징을 재구성합니다. 비디오 향상과 함께 사용하여 오래된 비디오를 리마스터할 수 있습니다.
사용자 정의 기능 처리	주문형 맞춤형 서비스를 유연하고 신속하게 구현하고, 개발을 가속화하고, 비용을 절감하고, 효율성을 높일 수 있도록 지원합니다.
비디오 암호화	비디오 보안을 보장하기 위해 HLS 표준 암호화를 통해 비디오 데이터를 암호화합니다.

기능	설명
비디오 태그	멀티 모달 정보 융합 및 정렬 기술을 기반으로 비디오의 시각, 장면, 동작, 사물 등을 분석하여 영상 콘텐츠를 정확하게 인식하고 다차원 콘텐츠 태그를 자동으로 출력합니다.
텍스트 음성 변환	고급 딥 러닝 기술을 통해 텍스트를 다양한 음성으로 자연스럽고 부드러운 음성으로 변환합니다.
음성/소리 분리	지정된 비디오(또는 오디오)에서 사람의 목소리와 배경음을 분리하여 다른 스타일의 후속 예술적 처리를 위한 오디오 자료를 생성합니다.
적응형 HLS muxing	단일 원본 비디오에서 다중 비트 레이트 적응 파일을 생성하여 비디오를 다양한 장치 및 네트워크 조건에 맞게 조정합니다.
HDR to SDR	출력 비디오의 이미지 세부 사항을 원본 비디오의 세부 사항에 최대한 가깝게 만들어 다양한 유형의 장치에 적용하고 이미지 왜곡 및 어두움을 방지합니다.
비디오 프레임 캡처	지정된 시점의 비디오의 프레임을 캡처합니다. 프레임 캡처 시작 시점, 프레임 캡처 간격, 캡처할 프레임 수, 출력 이미지 크기 및 형식을 사용자의 다양한 요구에 맞게 사용자 정의할 수 있습니다.
오디오/비디오 스플라이싱	비디오/오디오 파일의 처음 또는 끝에 비디오/오디오 세그먼트를 추가하여 새 파일을 생성합니다.
오디오/비디오 세분화	비디오/오디오 파일을 여러 세그먼트로 나눕니다.
동영상을 애니메이션 이미지로 변환	동영상 파일을 애니메이션 이미지 파일로 변환합니다. 변환을 위한 비디오 세그먼트, 프레임 샘플링 방법, 출력 애니메이션 이미지의 프레임 속도, 크기 및 형식을 지정하여 다양한 요구 사항을 충족할 수 있습니다.
스마트 썸네일	Tencent Cloud의 고급 AI 기술로 비디오 콘텐츠를 이해하여 비디오 프레임의 품질, 밝기 및 콘텐츠 관련성을 지능적으로 분석합니다. 그런 다음 최적의 프레임을 추출하여 썸네일을 생성하여 콘텐츠를 더욱 매력적으로 만듭니다.
비디오 메타데이터 수집	비디오 파일의 인코딩 형식, 코덱, 픽셀 형식, 지속 시간, 비트 레이트, 프레임 레이트, 너비 및 높이, 오디오 파일의 비트 레이트, 샘플 형식, 샘플링 레이트, 채널 수, 재생 시간과 같은 COS에 저장된 비디오, 오디오 및 자막과 같은 미디어 파일의 메타데이터를 가져옵니다. 이는 다양한 미디어 정보에 대한 요구 사항을 충족하는 데 도움이 됩니다.

적용 시나리오

다중 장치 적응성

콘텐츠 플랫폼은 일반적으로 여러 유형의 장치를 대상으로 하기 때문에 다양한 사용자에게 다양한 형식의 미디어 파일을 제공해야 합니다. 오디오/비디오 트랜스 코딩 기능은 대부분의 트랜스 코딩 요구 사항을 충족하며 압축 효율성을 높이고 파일 크기를 줄이기 위해 다양한 압축 기능을 제공합니다. 이를 통해 랙, 저장 공간 사용량 및 트래픽 요금을 낮춥니다.

비디오 플랫폼

기존 비디오 플랫폼의 경우 리뷰어가 비디오를 시청한 다음 수동으로 썸네일을 선택해야 하므로 공수가 많이 들고 비디오 릴리스가 느려집니다.

스마트 썸네일 기능은 가장 눈에 띄는 프레임을 썸네일로 빠르게 선택할 수 있어 공수를 절약하고 비디오 릴리스를 가속화할 수 있습니다.

비디오를 애니메이션 이미지로 변환하는 기능을 사용하면 비디오 플랫폼에서 비디오의 하이라이트를 선택하여 비디오 미리보기용 애니메이션 이미지로 변환할 수 있으므로 사용자가 재생하지 않고도 비디오를 살짝 볼 수 있습니다. 기존의 정적 비디오 썸네일과 비교할 때 애니메이션 이미지 썸네일은 클릭률과 비디오 재생을 증가시킵니다.

사용 방법

작업 구성 또는 Configuring Workflow를 통해 미디어 처리 기능을 사용할 수 있습니다. 효율성을 높이고 반복 작업을 줄이기 위해 오디오/비디오 트랜스 코딩, 오디오/비디오 스플라이싱, 비디오 프레임 캡처, 비디오를 애니메이션 이미지로 변환하는 기능에 대해 작업 또는 워크플로를 생성할 때 템플릿을 지정할 수 있습니다. 템플릿 페이지는 System Templates, Custom Templates를 참고하십시오.

작업

COS에 저장된 기존 데이터에 대한 미디어 처리 작업을 생성할 수 있습니다.

작업 관리

- 콘솔: 작업 구성의 안내에 따라 COS 콘솔에서 시각적으로 작업을 생성할 수 있습니다.
- API: Job APIs 에 안내된 대로 API를 통해 미디어 처리 작업을 생성, 삭제, 쿼리 및 검색할 수 있습니다.

워크플로

미디어 처리 워크플로를 사용하면 필요에 따라 오디오/비디오 처리 흐름을 빠르고 유연하게 생성할 수 있습니다. 워크플로는 입력 버킷의 경로에 바인딩됩니다. 파일이 경로에 **업로드**되면 미디어 워크플로가 **자동으로 트리거**되어 지정된 처리 작업을 수행하고 처리 결과가 출력 버킷의 지정된 경로에 자동으로 저장됩니다. 워크플로에서 **오디오/비디**

오 스플라이싱, **오디오/비디오 트랜스 코딩**, **비디오 프레임 캡처**, **비디오를 애니메이션 이미지로 변환** 및 **스마트 썸네일** 작업을 설정할 수 있습니다.

워크플로 관리

- 콘솔: Configuring Workflow의 안내에 따라 COS 콘솔에서 시각적으로 워크플로를 생성할 수 있습니다.
- API: API 문서에 안내된 대로 API를 통해 미디어 처리 워크플로를 생성, 삭제, 쿼리 및 검색할 수 있습니다.

파일 처리 개요

최종 업데이트 날짜: : 2023-04-27 15:27:23

개요

COS에 저장된 모든 파일에 대해 해시 계산, 압축 해제, 압축 및 패키징과 같은 파일 처리 기능이 제공됩니다. 현재 다음과 같은 파일 처리 기능이 지원됩니다.

기능	설명
해시 계산	파일 해시를 계산합니다. 현재 MD5, SHA1 및 SHA256 해시 계산 알고리즘이 지원됩니다. 파일 크기 제한: • 동기화 요청: 128MB 미만. • 비동기 요청: 50GB 미만.
파일 압축 해제	클라우드에서 .zip, tar, .gz 또는 .7zip 패키지의 압축을 해제하고 추출된 파일을 COS에 덤프합니다. 파일 크기 제한: 5TB 미만.
다중 파일 압축	여러 파일을 zip, tar 또는 tar.gz 형식으로 압축합니다. 파일 제한: 총 50GB 미만의 파일을 최대 10000개까지 압축할 수 있습니다.

설명 :

- 파일 처리 서비스는 CI에서 제공 및 과금합니다. 과금에 대한 자세한 내용은 File Processing Fees를 참고하십시오.

- 현재 파일 처리는 베이징, 상하이, 광저우, 청두, 중국홍콩, 싱가포르, 실리콘밸리 리전에서 지원됩니다.

적용 시나리오

데이터 검증

파일 해시 계산 기능을 사용하여 데이터 일관성을 빠르게 확인할 수 있습니다.

일상 툴

온클라우드 PaaS 파일 압축 및 압축 해제 기능이 제공되어, 압축 해제 후 파일을 온라인에서 미리 볼 수 있습니다. 사용자에게 더 포괄적인 온라인 미리보기 시나리오를 제공합니다.

사용 방법

파일 처리 기능은 CI에서 제공하므로 먼저 여기를 클릭하여 CI를 활성화해야 합니다.

CI를 활성화한 후 COS 콘솔에서 파일 처리를 활성화한 다음 콘솔 또는 API를 통해 기능을 사용할 수 있습니다.

COS 콘솔 사용

파일 목록을 통해 사용

버킷 파일 목록에서 **추가 작업**을 클릭하면 파일에 대한 해시 계산과 같은 파일 처리 작업을 수행할 수 있습니다.

태스크를 통해 사용

태스크 및 워크플로를 통해 파일 처리 작업을 수행할 수 있습니다.

> 주의 :
> 현재 이 기능은 워크플로가 아닌 태스크를 통해서만 사용할 수 있습니다.

콘텐츠 조정 개요

최종 업데이트 날짜: : 2024-03-12 15:46:48

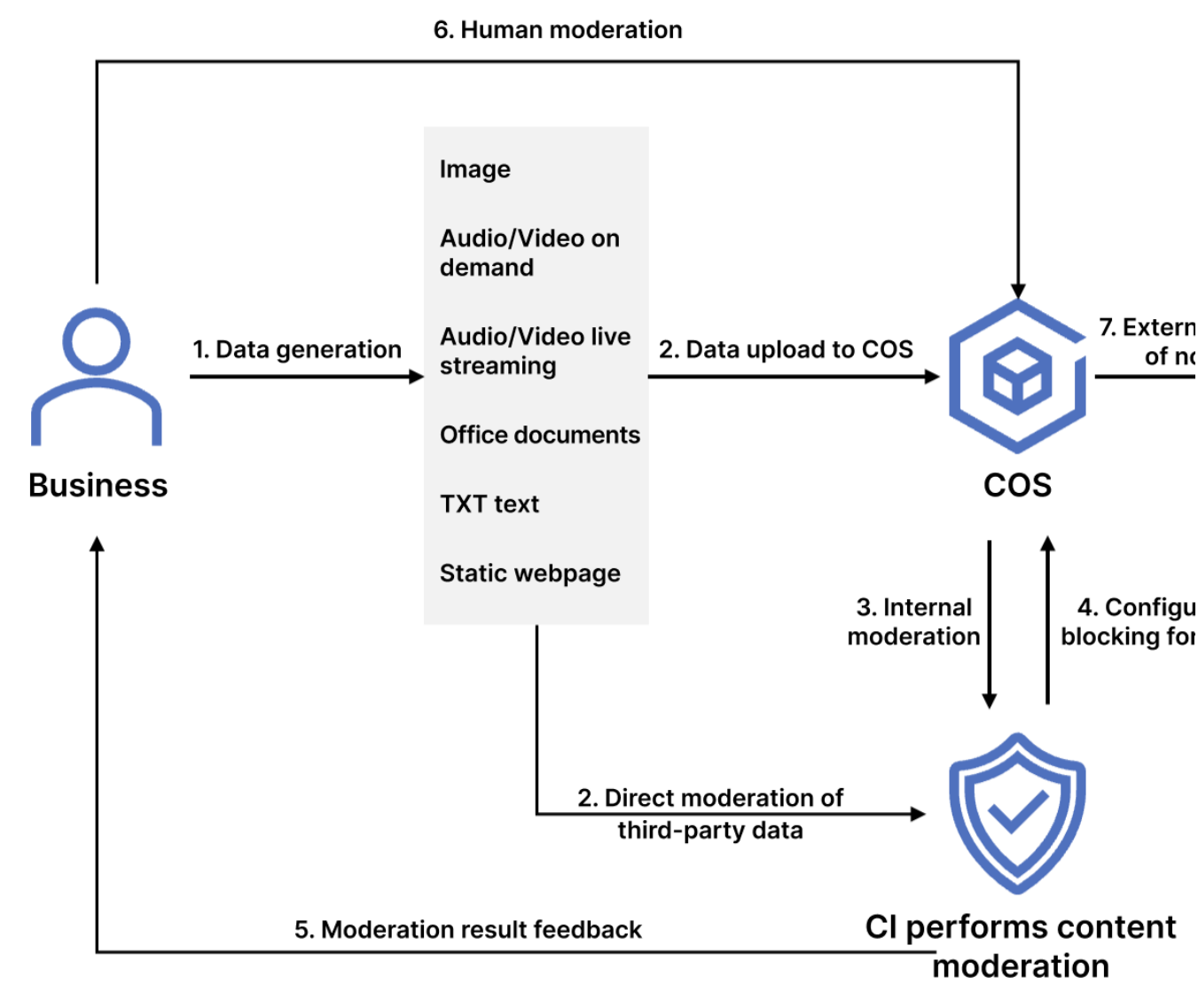
개요

COS 콘텐츠 조정 서비스는 이미지, 비디오, 오디오, 텍스트, 문서 및 웹 페이지의 멀티미디어 콘텐츠를 지능적으로 조정합니다. 선정적이고 저속한 콘텐츠, 규정 위반 콘텐츠, 역겹고 반감을 일으키는 콘텐츠와 같은 비준수 콘텐츠를 효과적으로 식별하여 운영 상의 위험을 방지할 수 있습니다. 현재 다음 형식의 파일을 조정할 수 있습니다.

파일 유형	제한
이미지	png, jpeg, jpg, bmp, webp, gif, heif, heic 포맷을 지원합니다. 32MB 이하의 이미지 조정을 지원하며, 5MB를 초과하는 이미지의 경우, 요청을 호출할 때 large-image-detect 파라미터를 사용해야 합니다. 이미지 해상도는 20 x 20 이상, 40000 x 40000(동기 조정은 10000 x 10000 이하) 이하이며, 식별 효과를 고려하여 256x256 이상의 해상도를 권장합니다.
비디오	mp4, avi, mkv, wmv, rmvb, flv, m3u8 형식이 지원됩니다. 동영상 크기는 5GB를 초과할 수 없으며 캡처된 프레임 수는 1만을 초과할 수 없습니다.
오디오	오디오 포맷: 현재 mp3, wav, aac, flac, amr, 3gp, m4a, wma, ogg, ape를 지원합니다. 오디오 비트레이트: 128Kbps - 256Kbps. 오디오 크기: 파일은 600MB 이하입니다. 최장 시간: 3시간. 오디오 파일 콘텐츠 언어 지원: 표준 중국어, 영어, 광둥어. 비디오 파일로 입력할 때, 비디오 파일의 사운드 트랙 분리와 오디오 콘텐츠에 대한 개별 조정을 지원합니다.
텍스트	먼저 base64로 인코딩해야 하며, 텍스트 인코딩 전의 원본 길이는 10000개 utf8 인코딩 문자를 초과해서는 안 됩니다. html, txt 포맷을 지원하고, UTF8 인코딩과 GBK 인코딩만 지원하며, 파일 크기는 1MB를 초과하지 않습니다. 언어는 중국어, 영어, 아라비아 숫자에 대한 검출을 지원합니다.
라이브 스트림	라이브 스트림 시간 지원: 5시간 이내. 지원되는 라이브 스트림 미디어 프로토콜: rmtp, hls, http, https 등과 같은 주류 프로토콜. 라이브 스트림 해상도 지원: 최대 1920x1080(1080p). 기본 동시 조정 경로 수 제한: 10개, 동시 조정 경로 수를 초과하는 경우 API가 오류를 반환합니다.
문서	문서 처리 서비스를 통해 조정을 위해 이미지로 변환됩니다. 현재 pdf, ppt, excel 등 수십 개의 문서 형식을 지원합니다. 자세한 내용은 문서 미리보기 개요를 참고하십시오.
웹페이지	시스템은 딥러닝 기술을 기반으로 웹페이지 파일을 자동으로 감지하고 OCR, 객체 감지(객체, 광고 로고, QR 코드 등), 이미지 인식 차원에서 비준수 콘텐츠를 인식할 수 있습니다.

설명 :

콘텐츠 조정은 유료 서비스이며, 요금은 CI에서 청구합니다.



조정 서비스를 활성화한 후 다음 작업을 수행할 수 있습니다.

자동 조정을 구성하여 버킷에 업로드된 데이터를 자동으로 조정하고 비준수 데이터를 차단합니다

API를 호출하여 타사 데이터를 조정합니다

버킷의 기존 데이터에 대해 일회성 일괄 조정을 수행합니다

사용 사례

이 기능은 소셜 네트워크, 전자 상거래, 광고, 게임 등 분야에 적합합니다. 상기 유형의 파일에서 음란물, 불법, 광고 콘텐츠를 확인할 수 있습니다.

사용 가능한 리전

다음 리전이 지원됩니다.

지역		지역 약칭
중국 대륙	베이징	ap-beijing
	난징	ap-nanjing
	상하이	ap-shanghai
	광저우	ap-guangzhou
	청두	ap-chengdu
	충칭	ap-chongqing
중국 홍콩 및 해외	중국 홍콩	ap-hongkong
	싱가포르	ap-singapore
	봄베이	ap-mumbai
	프랑크푸르트	eu-frankfurt

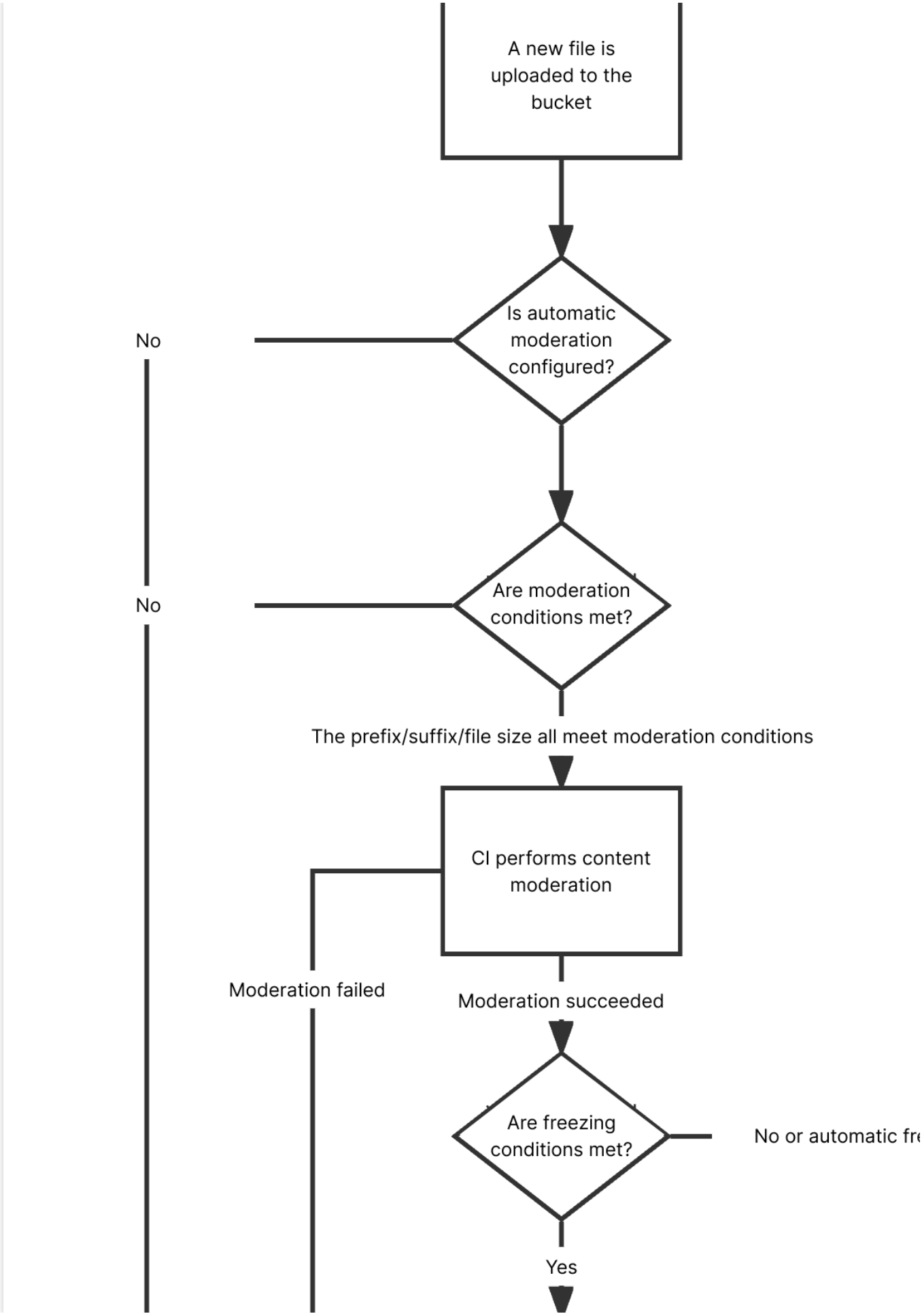
다른 리전에서 콘텐츠 조정 기능을 사용하려면 티켓 제출을 통해 문의하십시오.

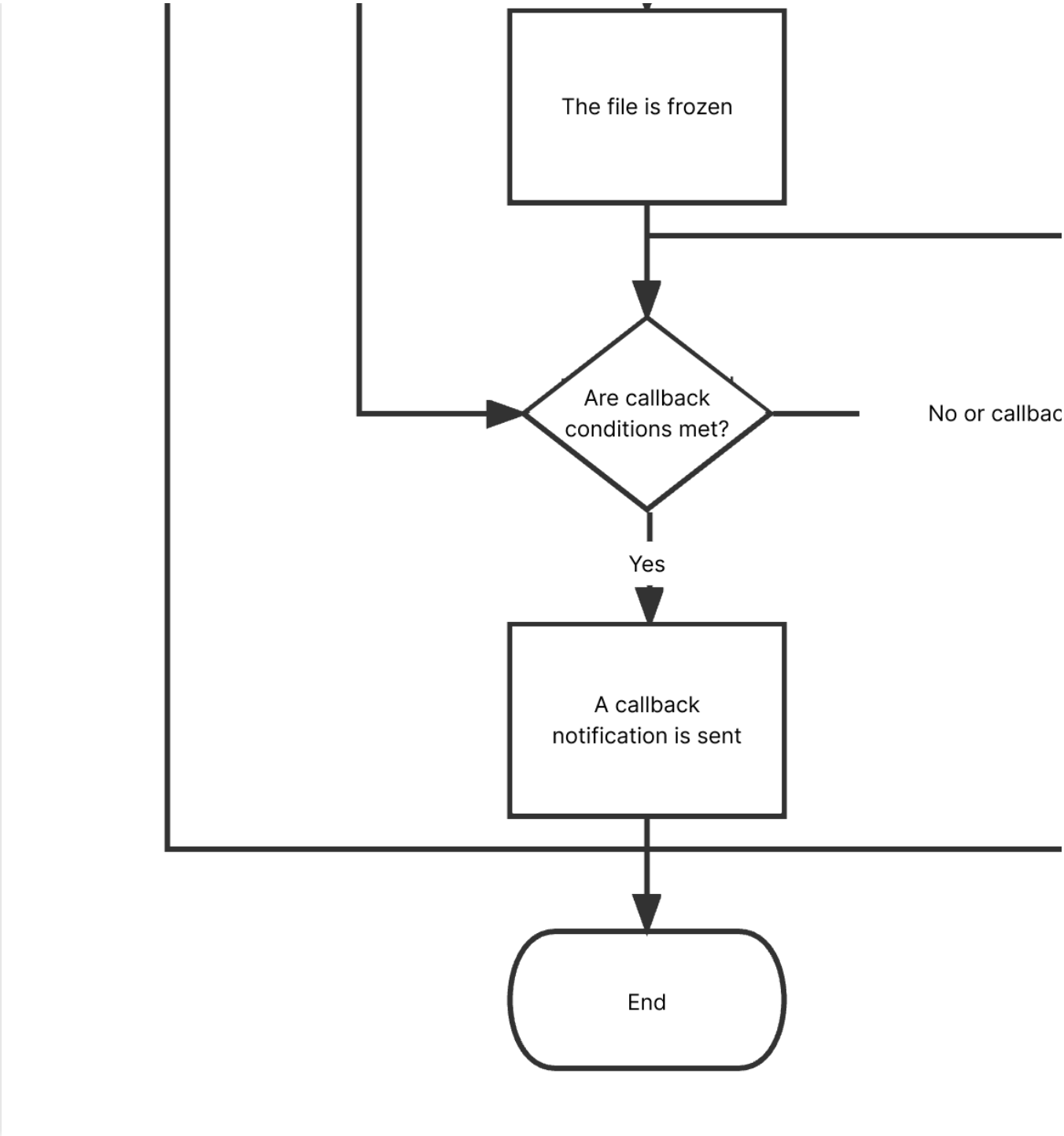
사용 방법

COS 콘솔 사용

자동 조정

COS 콘솔에서 자동 조정 서비스를 활성화하여 새로 업로드된 이미지, 비디오, 오디오, 파일, 문서 및 웹 페이지를 자동으로 조정할 수 있습니다. 자세한 내용은 자동 조정을 참고하십시오.





기존 데이터 조정

COS 콘솔에서 기록 데이터 조정 서비스를 활성화하여 버킷에 이미 저장된 이미지, 비디오, 오디오, 텍스트, 문서 및 웹 페이지에 대해 일회성 일괄 조정을 수행할 수 있습니다.

API 사용

API를 사용하여 이미지, 비디오, 오디오, 텍스트, 문서 및 웹 페이지의 콘텐츠를 조정할 수 있습니다. 자세한 내용은 다음 API 문서를 참고하십시오.

Single Image Moderation

Submitting Video Moderation Job
Submitting Audio Moderation Job
텍스트 조정
문서 조정
웹 페이지 조정
라이브 스트림 조정
조정 후 민감한 데이터로 확인된 파일의 경우 다음 방법 중 하나를 선택하여 처리하는 것이 좋습니다.
사용자가 공중망을 통해 익명으로 액세스하지 못하도록 파일의 액세스 권한을 비공개 읽기로 변경합니다. 자세한 내용은 PUT Object acl을 참고하십시오.
파일을 백업 디렉터리로 이동합니다. 원본 파일을 지정된 디렉터리에 복사한 후 원본 파일을 삭제하여 파일을 이동합니다. 자세한 내용은 PUT Object - Copy 및 DELETE Object를 참고하십시오.
파일을 삭제합니다. 자세한 내용은 DELETE Object를 참고하십시오.

SDK 사용

또한 다양한 프로그래밍 언어용 SDK를 사용하여 이미지, 비디오, 오디오, 텍스트, 문서 및 웹 페이지의 콘텐츠를 조정할 수 있습니다. 자세한 내용은 다음 SDK 문서를 참고하십시오.

SDK	통합 가이드
Android SDK	콘텐츠 조정
C SDK	콘텐츠 조정
.NET(C#) SDK	콘텐츠 조정
Go SDK	콘텐츠 조정
iOS SDK	콘텐츠 조정
Java SDK	콘텐츠 조정
JavaScript SDK	콘텐츠 조정
PHP SDK	콘텐츠 조정
Python SDK	콘텐츠 조정

모니터링 및 알람

최종 업데이트 날짜: : 2022-11-14 14:28:36

개요

COS의 읽기/쓰기 요청량, 트래픽 등의 데이터는 Cloud Monitor를 기반으로 통계 및 표시합니다. COS 콘솔 또는 클라우드 모니터링의 콘솔에서 COS의 읽기/쓰기 요청량, 트래픽 등 자세한 모니터링 데이터를 확인할 수 있습니다.

설명 :

- 본 문서는 COS 콘솔에서 통계를 얻는 방법을 설명합니다. Cloud Monitor API를 호출하여 더 자세한 데이터를 얻을 수 있습니다. 자세한 내용은 Cloud Monitor 제품 문서를 참고하십시오.
- 현재 CM에 보고된 모든 메트릭은 모든 COS 리전을 지원합니다. 자세한 내용은 리전 및 액세스 도메인을 참고하십시오.

기본 기능

Cloud Monitor는 COS에 다음과 같은 게이트를 제공하여 모니터링과 알람 기능을 구현합니다.

모듈	설명	주요 기능
모니터링 개요	제품의 현재 상태 표시	전체 현황, 알람 현황, 전체 모니터링 정보 목록 제공
알람 관리	알람 관리 및 설정 지원	COS에 신규 추가된 알람 정책, 사용자 정의 정보, 트리거 조건 템플릿 지원
모니터링 플랫폼	트래픽 모니터링 및 사용자 정의한 모니터링 지표 데이터 조회	사용자 전체 대역폭 정보 조회, 사전에 모니터링 지표 및 보고 데이터 사용자 정의
클라우드 서비스 모니터링	COS의 버킷 모니터링 도표 조회	현재 각 버킷의 읽기/쓰기 요청량, 트래픽 등 모니터링 도표 및 데이터 조회

시나리오

- **일상 관리 시나리오**: Cloud Monitor 콘솔에 로그인하여 COS의 실행 상태를 실시간으로 확인합니다.

- **이상 경고 처리 시나리오**: 모니터링 데이터가 알람 임계값에 도달하는 경우, 알람 정보를 발송하여 사용자가 빠르게 이상 경고 알림을 수신하고 원인을 찾아 즉시 처리할 수 있도록 합니다.

콘솔에서 설정 및 조회

Cloud Monitor 콘솔을 통해 COS에 알람 정책을 생성하고, 모니터링 지표가 설정값에 이르면 알람을 수신합니다. 작업 관련 가이드는 모니터링 알람 설정을 참고하십시오.

모니터링 데이터를 조회하려면, **클라우드 서비스 모니터링 > COS** 페이지에서 모든 버킷의 모니터링 데이터, 상태, 알람 정책 수를 확인할 수 있습니다. 또한, COS 콘솔에서도 조회할 수 있으며, 작업 가이드는 데이터 개요 조회 및 데이터 모니터링 조회를 참고하십시오.

인터페이스로 호출

호출 인터페이스를 통해 COS의 모니터링 데이터를 조회할 수 있으며, COS의 모니터링 항목은 다음과 같습니다. COS의 모니터링 인터페이스에 대한 자세한 내용은 COS 모니터링 지표 문서를 참고하십시오.

모니터링 지표

설명：
COS는 범용 리전을 사용합니다. 따라서 버킷이 어떤 리전에 소속되어 있든 COS 모니터링 지표 데이터 풀링 시, Region은 항상 '광저우' 리전으로 선택하십시오.

- API Explorer를 사용해 데이터 풀링 시, Region 필드는 항상 '화남 리전(광저우)'으로 선택하십시오.
- SDK를 사용해 데이터 풀링 시, Region 필드는 항상 'ap-guangzhou'로 입력하십시오.

요청 관련

지표 이름(영어)	지표 이름(한글)	지표의 의미	단위	차원
StdReadRequests	스탠다드 스토리지 읽기 요청	스탠다드 스토리지 유형의 읽기 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket
StdWriteRequests	스탠다드 스토리지 쓰기 요청	스탠다드 스토리지 유형의 쓰기 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket

지표 이름(영어)	지표 이름(한글)	지표의 의미	단위	차원
IaReadRequests	스탠다드IA 스토리지 읽기 요청	스탠다드IA 스토리지 유형의 읽기 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket
IaWriteRequests	스탠다드IA 스토리지 쓰기 요청	스탠다드IA 스토리지 쓰기 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket
DeepArcReadRequests	DEEP ARCHIVE 읽기 요청	DEEP ARCHIVE 유형의 읽기 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket
DeepArcWriteRequests	DEEP ARCHIVE 쓰기 요청	DEEP ARCHIVE 유형의 쓰기 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket
ItReadRequests	INTELLIGENT TIERING 읽기 요청	INTELLIGENT TIERING 유형의 읽기 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket
ItWriteRequests	INTELLIGENT TIERING 쓰기 요청	INTELLIGENT TIERING 유형의 쓰기 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket
TotalRequests	총 요청 수	모든 스토리지 유형의 총 읽기/쓰기 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket
GetRequests	GET 관련 총 요청 수	모든 스토리지 유형의 GET 관련 총 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket
PutRequests	PUT 관련 총 요청 수	모든 스토리지 유형의 PUT 관련 총 요청 횟수. 요청 횟수는 요청 명령 발송 횟수에 따라 계산됨	회	appid, bucket

스토리지 관련

지표 이름(영어)	지표 이름(한글)	단위	차원
StdStorage	스탠다드 스토리지 - 스토리지 용량	MB	appid, bucket

지표 이름(영어)	지표 이름(한글)	단위	차원
SiaStorage	스탠다드IA 스토리지 - 스토리지 용량	MB	appid, bucket
ItFreqStorage	INTELLIGENT TIERING - 고빈도 레이어 스토리지 용량	MB	appid, bucket
ItInfreqStorage	INTELLIGENT TIERING - 저빈도 레이어 스토리지 용량	MB	appid, bucket
ArcStorage	ARCHIVE - 스토리지 용량	MB	appid, bucket
DeepArcStorage	DEEP ARCHIVE - 스토리지 용량	MB	appid, bucket
StdObjectNumber	스탠다드 스토리지 - 객체 수	개	appid, bucket
IaObjectNumber	스탠다드IA 스토리지 - 객체 수	개	appid, bucket
ItFreqObjectNumber	INTELLIGENT TIERING_고빈도 레이어 객체 수	개	appid, bucket
ItInfreqObjectNumber	INTELLIGENT TIERING_저빈도 레이어 객체 수	개	appid, bucket
ArcObjectNumber	ARCHIVE 객체 수	개	appid, bucket
DeepArcObjectNumber	DEEP ARCHIVE 객체 수	개	appid, bucket
StdMultipartNumber	스탠다드 스토리지 - 파일 조각 수	개	appid, bucket
IaMultipartNumber	스탠다드IA 스토리지 - 파일 조각 수	개	appid, bucket
ItFrequentMultipartNumber	INTELLIGENT TIERING - 고빈도 파일 조각 수	개	appid, bucket
ArcMultipartNumber	ARCHIVE - 파일 조각 수	개	appid, bucket

지표 이름(영어)	지표 이름(한글)	단위	차원
DeepArcMultipartNumber	DEEP ARCHIVE - 파일 조각 수	개	appid, bucket

트래픽 관련

지표 이름(영어)	지표 이름(한글)	지표의 의미	단위	차원
InternetTraffic	공인 네트워크 다운스트림 트래픽	인터넷을 통해 데이터를 COS에서 클라이언트로 다운로드할 때 발생하는 트래픽	B	appid, bucket
InternetTrafficUp	공인 네트워크 업스트림 트래픽	인터넷을 통해 데이터를 클라이언트에서 COS로 업로드할 때 발생하는 트래픽	B	appid, bucket
InternalTraffic	내부 네트워크 다운스트림 트래픽	Tencent Cloud 내부 네트워크를 통해 데이터를 COS에서 클라이언트로 다운로드할 때 발생하는 트래픽	B	appid, bucket
InternalTrafficUp	내부 네트워크 업스트림 트래픽	Tencent Cloud 내부 네트워크를 통해 데이터를 클라이언트에서 COS로 업로드할 때 발생하는 트래픽	B	appid, bucket
CdnOriginTraffic	CDN Origin-pull 트래픽	데이터를 COS에서 Tencent Cloud CDN 엣지 노드로 전송할 때 발생하는 트래픽	B	appid, bucket
InboundTraffic	공인 네트워크, 내부 네트워크 업로드 총 트래픽	인터넷 및 Tencent Cloud 내부 네트워크를 통해 데이터를 클라이언트에서 COS로 업로드할 때 발생하는 트래픽	B	appid, bucket
CrossRegionReplicationTraffic	리전 간 복제 트래픽	데이터를 한 리전의 버킷에서 다른 리전의 버킷으로 전송할 때 발생하는 트래픽	B	appid, bucket

반환 코드 관련

지표 이름(영어)	지표 이름(한글)	지표의 의미	단위	차원

지표 이름(영어)	지표 이름(한글)	지표의 의미	단위	차원
2xxResponse	2xx 상태 코드	2xx 상태 코드가 반환된 요청 횟수	회	appid, bucket
3xxResponse	3xx 상태 코드	3xx 상태 코드가 반환된 요청 횟수	회	appid, bucket
4xxResponse	4xx 상태 코드	4xx 상태 코드가 반환된 요청 횟수	회	appid, bucket
5xxResponse	5xx 상태 코드	5xx 상태 코드가 반환된 요청 횟수	회	appid, bucket
2xxResponseRate	2xx 상태 코드 비율	총 요청 횟수 중 2xx 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket
3xxResponseRate	3xx 상태 코드 비율	총 요청 횟수 중 3xx 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket
4xxResponseRate	4xx 상태 코드 비율	총 요청 횟수 중 4xx 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket
5xxResponseRate	5xx 상태 코드 비율	총 요청 횟수 중 5xx 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket
400Response	400 상태 코드	400 상태 코드가 반환된 요청 횟수	회	appid, bucket
403Response	403 상태 코드	403 상태 코드가 반환된 요청 횟수	회	appid, bucket
404Response	404 상태 코드	404 상태 코드가 반환된 요청 횟수	회	appid, bucket
400ResponseRate	400 상태 코드 비율	총 요청 횟수 중 400 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket
403ResponseRate	403 상태 코드 비율	총 요청 횟수 중 403 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket
404ResponseRate	404 상태 코드 비율	총 요청 횟수 중 404 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket
500ResponseRate	500 상태 코드 비율	총 요청 횟수 중 500 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket

지표 이름(영어)	지표 이름(한글)	지표의 의미	단위	차원
501ResponseRate	501 상태 코드 비율	총 요청 횟수 중 501 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket
502ResponseRate	502 상태 코드 비율	총 요청 횟수 중 502 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket
503ResponseRate	503 상태 코드 비율	총 요청 횟수 중 503 상태 코드가 반환된 요청 횟수의 비율	%	appid, bucket

설명：

1. 3xx, 4xx, 5xx 상태 코드에 관한 자세한 내용은 Error Codes를 참고하십시오.
2. 각 지표의 통계 단위(Period)는 모두 상이합니다. DescribeBaseMetrics 인터페이스를 통해 각 지표가 지원하는 통계 단위를 확인할 수 있습니다.

데이터 읽기 관련

지표 이름(영어)	지표 이름(한글)	지표의 의미	단위	차원
StdRetrieval	표준 데이터 읽기량	표준 데이터를 읽을 때 발생하는 트래픽. 스탠다드 스토리지의 공인 네트워크 다운스트림 트래픽, 내부 네트워크 다운스트림 트래픽, CDN Origin-pull 트래픽의 총합	B	appid, bucket
IaRetrieval	스탠다드IA 데이터 읽기량	스탠다드IA 데이터를 읽을 때 발생하는 트래픽. 스탠다드IA 스토리지의 공인 네트워크 다운스트림 트래픽, 내부 네트워크 다운스트림 트래픽, CDN Origin-pull 트래픽의 총합	B	appid, bucket

데이터 처리 클래스

API를 호출하여 CI(Cloud Infinite)의 모니터링 데이터를 볼 수 있으며, CI의 모니터링 API에 대한 자세한 내용은 CI 모니터링 지표 문서를 참고하십시오.

각 차원의 해당 매개변수 개요

매개변수 이름	차원 이름	차원 설명	포맷
&Instances.N.Dimensions.0.Name	appid	루트 계정 APPID의 차원 이름	String 유형의 차원 이름 입력: appid
&Instances.N.Dimensions.0.Value	appid	루트 계정의 구체적인 APPID	루트 계정 APPID 입력. 예: 1250000000
&Instances.N.Dimensions.1.Name	bucket	버킷 차원 이름	String 유형의 차원 이름 입력: bucket
&Instances.N.Dimensions.1.Value	bucket	버킷의 구체적인 이름	버킷의 구체적인 이름 입력. 예: examplebucket-1250000000

입력 매개변수 설명

COS 모니터링 데이터를 조회합니다. 입력 매개변수 값은 다음과 같습니다.

&Namespace=QCE/COS
&Instances.N.Dimensions.0.Name=appid
&Instances.N.Dimensions.0.Value=루트 계정의 APPID
&Instances.N.Dimensions.1.Name=bucket
&Instances.N.Dimensions.1.Value=버킷 이름

모니터링 설명

- **모니터링 간격**: Cloud Monitor는 실시간, 최근 24시간, 최근 7일, 사용자 정의 날짜 등 여러 가지 통계 구간에 따라 모니터링 데이터를 제공하며, 시간은 1분, 5분, 1시간, 1일로 세분화됩니다.
- **데이터 스토리지**: 1분 단위 모니터링 데이터, 15일 저장 / 5분 단위 모니터링 데이터, 31일 저장 / 1시간 단위 모니터링 데이터, 93일 저장 / 1일 단위 모니터링 데이터, 186일 저장을 지원합니다.
- **알람 표시**: Cloud Monitor는 COS의 모니터링 데이터를 통합하여 보기 쉬운 도표 형태로 표시합니다. 제품에 미리 정의된 알람 지표에 따라 알람을 보낼 수 있어 사용자가 전체적인 운영 상태를 파악할 수 있습니다.
- **알람 설정**: 모니터링 지표 임계값을 설정하고, 모니터링 데이터가 알람 조건을 충족하면 Cloud Monitor가 즉시 알람 정보를 관련 그룹에 전송합니다. 자세한 내용은 Alarm Overview 및 모니터링 알람 설정을 참고하십시오.