一、MinIO介绍

1.1、MinIO简介

无论是在技术上还是在操作上，MinIO使用和部署都非常简单。节点上只需一个二进制文件，可支持各种平台。

MinIO 是世界上最快的对象存储，没有之一。在 32 个 NVMe 驱动器节点和 100Gbe 网络上发布的 GET/PUT 结果超过 325 GiB/秒和 165 GiB/秒。

Minio 除了直接作为对象存储使用，还可以作为云上对象存储服务的网关层，无缝对接到 Amazon S3、 MicroSoft Azure。

通过原生 Kubernetes 运营商集成，MinIO 支持公共云、私有云和边缘云上所有主要的 Kubernetes 发行版。

1.2、MinIO的基础概念

Object：存储到 Minio 的基本对象，如文件、字节流，Anything...
Bucket：用来存储 Object 的逻辑空间。每个 Bucket 之间的数据是相互隔离的。对于客户端而 言，就相当于一个存放文件的顶层文件夹。
Drive：即存储数据的磁盘，在 MinIO 启动时，以参数的方式传入。Minio 中所有的对象数据都会 存储在 Drive 里。
Set ：即一组 Drive 的集合，分布式部署根据集群规模自动划分一个或多个 Set ，每个 Set 中的 Drive 分布在不同位置。一个对象存储在一个 Set 上。 （For example: {1...64} is divided into 4 sets each of size 16.）

- 一个对象存储在一个Set上
- 一个集群划分为多个Set
- 一个Set包含的Drive数量是固定的，默认由系统根据集群规模自动计算得出 一个SET中的Drive尽可能分布在不同的节点上

1.3、纠删码

MinIO 使用纠删码机制来保证高可靠性，使用 highwayhash 来处理数据损坏（ Bit Rot Protection ）。 关于纠删码，简单来说就是可以通过数学计算，把丢失的数据进行还原，它可以将n份原始数据，增加m 份数据，并能通过n+m份中的任意n份数据，还原为原始数据。即如果有任意小于等于m份的数据失效，仍然能通过剩下的数据还原出来。

Minio使用纠删码 erasure code 和校验和 checksum 来保护数据免受硬件故障和无声数据损坏。 即便丢失一半数量（N/2）的硬盘，仍然可以恢复数据。

纠删码是一种恢复丢失和损坏数据的数学算法， Minio采用Reed-Solomon code将对象拆分成N/2 数据和N/2 奇偶校验块。 这就意味着如果是12块盘，一个对象会被分成6个数据块、6个奇偶校验块，即使丢失任意6块盘（不管其是存放的数据块还是奇偶校验块），仍可以从剩下的盘中的数据进行恢复。

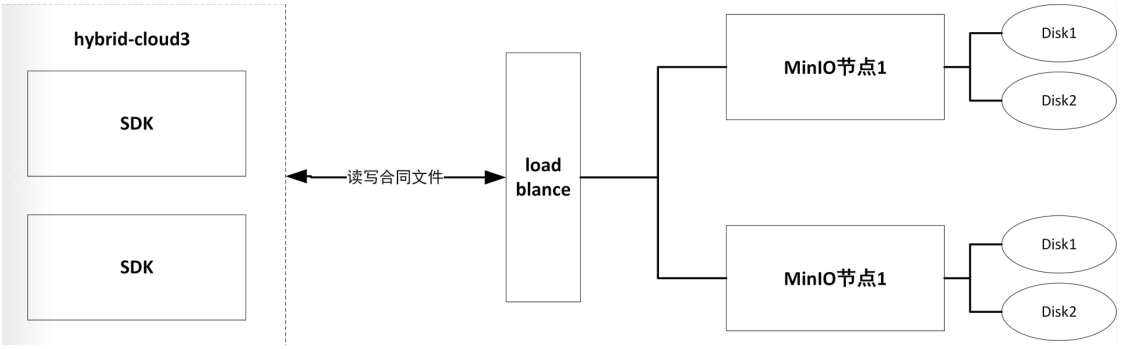
分布式Minio至少需要4个硬盘，使用分布式Minio自动引入了纠删码功能。

单机Minio服务存在单点故障，相反，如果是一个有N块硬盘的分布式Minio,只要有N/2硬盘在线，你的数据就是安全的。不过你需要至少有N/2+1个硬盘来创建新的对象。
例如，一个16节点的Minio集群，每个节点16块硬盘，就算8台服务器宕机，这个集群仍然是可读的，不 过你需要9台服务器才能写数据。

二、MinIO部署

2.1、部署环境（2节点，单节点2硬盘）

Hostname	IP	数据盘	备注
sdk-server	192.168.10.100	/	混合云服务、Nginx
minio-node-01	192.168.10.102	/minio_data1、/minio_data2	MioIO节点
minio-node-02	192.168.10.103	/minio_data1、/minio_data2	MioIO节点



2.2、部署MinIO

以下操作需在minio所有节点上执行相同操作

创建并进入MinIO程序目录

```
# mkdir -p  /data/minio/{run,log}
# cd /data/minio/run/
```

下载MinIO到当前目录:

```
# wget https://dl.min.io/server/minio/release/linux-amd64/minio ./
```

创建启动脚本

```
# cat >> /data/minio/run/run.sh <<"EOF"
#!/bin/bash
export MINIO_ROOT_USER=ssq
export MINIO_ROOT_PASSWORD=Bestsign@2019

/data/minio/run/minio server  --config-dir /etc/minio --address ":9000" --
console-address ":9001" \
http://192.168.10.10{2...3}/minio_data{1...2} >
/data/minio/log/minio_server.log

EOF
```

systemd配置文件minio.service

```
# cat >> /usr/lib/systemd/system/minio.service <<"EOF"
[Unit]
Description=Minio service
Documentation=https://docs.minio.io/

[Service]
WorkingDirectory=/data/minio/run/
ExecStart=/data/minio/run/run.sh

Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target

EOF
```

增加执行权限

```
# chmod +x /usr/lib/systemd/system/minio.service && chmod +x
/data/minio/run/minio && chmod +x /data/minio/run/run.sh
```

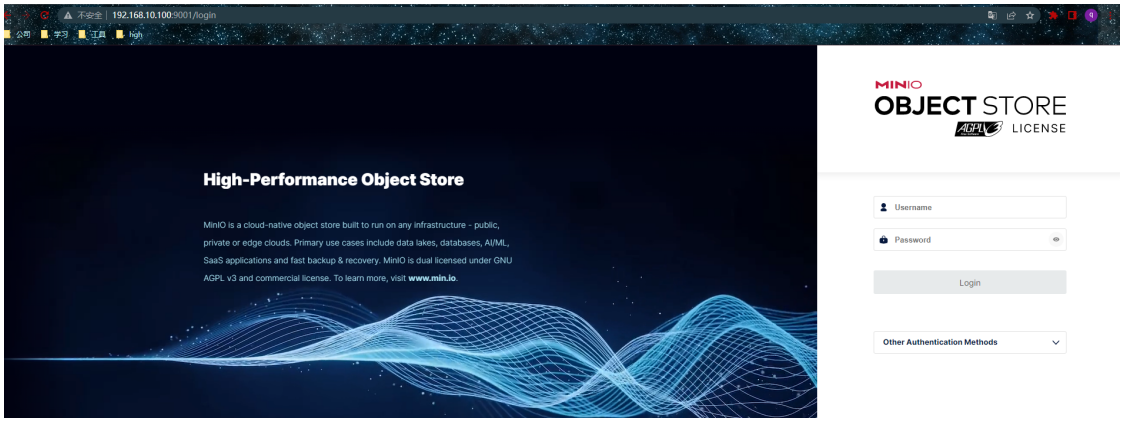
重新加载MinIO服务配置文件，并设置开机启动

```
# systemctl daemon-reload
# systemctl enable minio
```

所有节点配置完成后，依次启动MinIO

```
# systemctl start minio
```

任意节点通过浏览器访问：节点+9000 端口即可访问

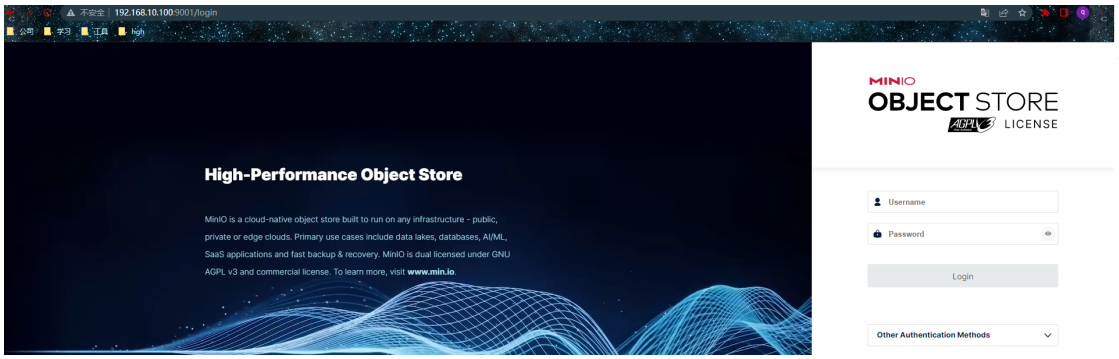


2.3、Nginx代理MinIO

```
server {
    listen       9000;
    server_name  localhost;
    # To allow special characters in headers
    ignore_invalid_headers off;
    # Allow any size file to be uploaded.
    # Set to a value such as 1000m; to restrict file size to a specific value
    client_max_body_size 0;
    # To disable buffering
    proxy_buffering off;

    location / {
         proxy_pass  http://miniostorage;
```

```
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-NginX-Proxy true;
        proxy_connect_timeout 300;
        # Default is HTTP/1, keepalive is only enabled in HTTP/1.1
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        chunked_transfer_encoding off;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "connection_upgrade";
    }
}
server {
    listen       9001;
    server_name  localhost;
    # To allow special characters in headers
    ignore_invalid_headers off;
    # Allow any size file to be uploaded.
    # Set to a value such as 1000m; to restrict file size to a specific value
    client_max_body_size 0;
    # To disable buffering
    proxy_buffering off;


    location / {
        proxy_pass  http://minioconsole;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-NginX-Proxy true;
        proxy_connect_timeout 300;
        # Default is HTTP/1, keepalive is only enabled in HTTP/1.1
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        chunked_transfer_encoding off;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "connection_upgrade";
    }
}

    upstream miniostorage {
```

```
        server 192.168.10.102:9000 weight=100;
        server 192.168.10.103:9000 weight=100;
    }

    upstream minioconsole {
        server 192.168.10.102:9001 weight=100;
        server 192.168.10.103:9001 weight=100;
    }
```



2.4、mc客户端使用

mc客户端下载

```
# wget http://dl.minio.org.cn/client/mc/release/linux-amd64/mc  ./
# chmod +x mc && mv mc /usr/local/sbin/
```

mc常用命令

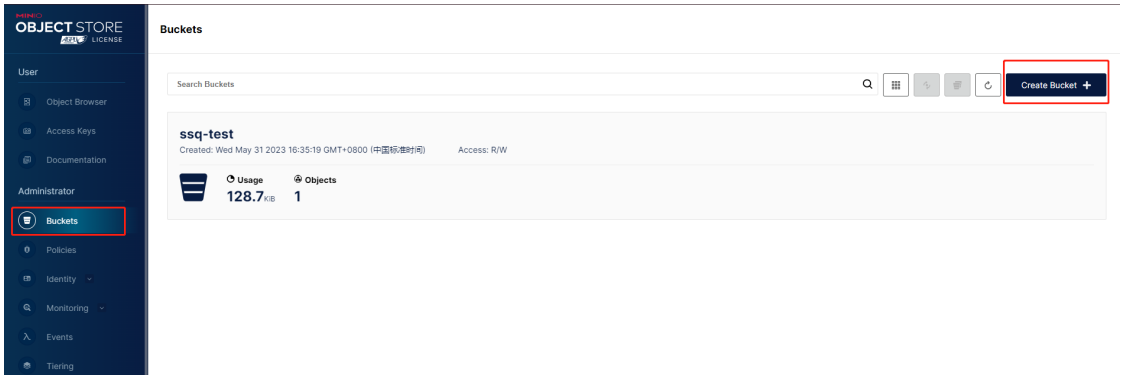
```
mc config host ls  # 查询mc host配置

mc config host add ssq-minio http://192.168.10.100:9000 sLOPBIbNOxq3i196
yuFO9matfokD22UnRF5V6XGDVRJMpHuH  #添加minio服务

mc config host remove ssq-minio 删除host

mc  ls  ssq-minio #列出存储桶和对象

mc ls ssq-minio/ssq-test/s3-
storage/data/1c83c0ac25d835b365d7b4321d668bd5a9e3b569

mc cp  ssq-minio/ssq-test/s3-
storage/data/1c83c0ac25d835b365d7b4321d668bd5a9e3b569  ./   #下载对象到指定文
件夹
```

```
mc cp  nginx-1.15.12.tar.gz ssq-minio/ssq-test/s3-storage/data/
#上传文件到指定桶的指定位置

mc mb ssq-minio/new-bucket      #创建桶

mc admin heal -r ssq-minio/ssq-test   #递归恢复存储桶和对象

mc cp -r ssq-minio/ssq-test/  ./          #递归下载指定存储桶到本地

mc cp -r s3-storage ssq-minio/new-bucket/        #递归上传对象到指定存储桶
```

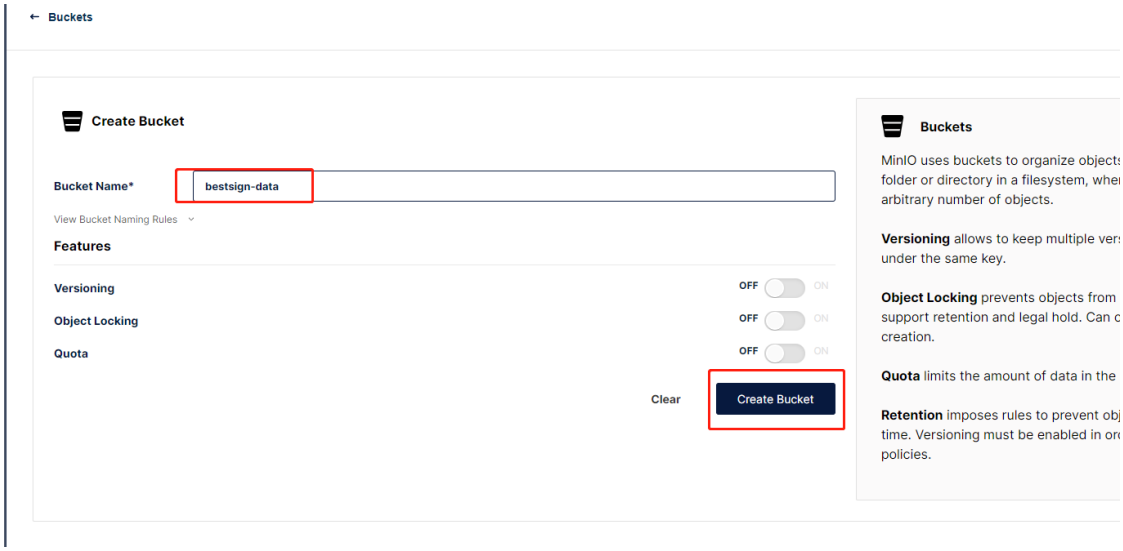
三、混3使用

3.1、创建Bucket

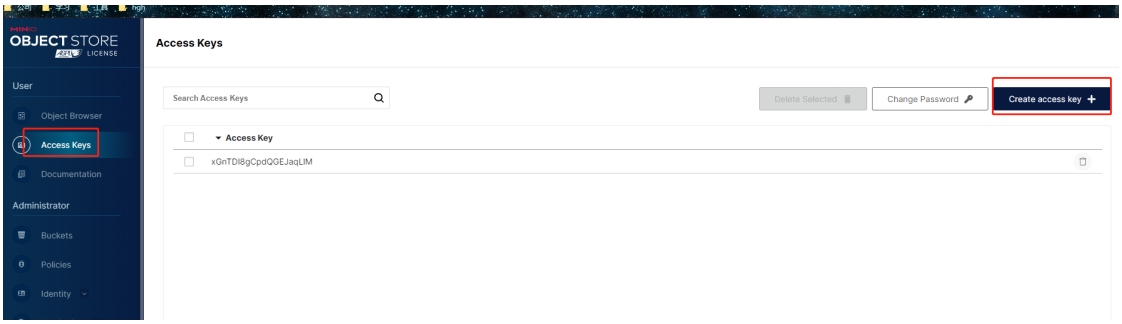
页面创建：



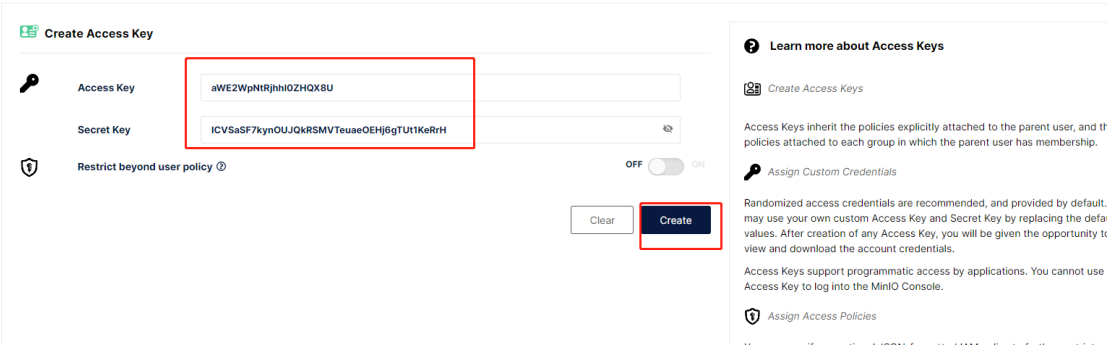
输入桶的名字即可创建



3.2、创建Access Key



自行保存 Access Key、Secret Key

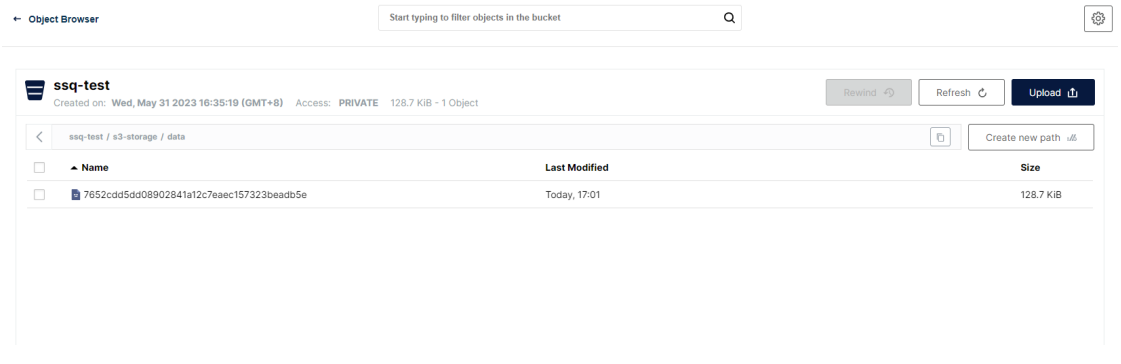


3.3、配置混3配置文件

application.properties 存储存储配置如下

```
storage.type=s3-customize
storage.pathStyleAccessEnabled=false
storage.endPoint=http://192.168.10.100:9000
storage.accessKey=xGnTDI8gCpdQGEJaqLlM
storage.accessKeySecret=MCNNrMsCG1s4v1PZdganKDHc6vXD5SLZ9jKh7voP
storage.bucket=ssq-test
storage.rootPath=/data
```

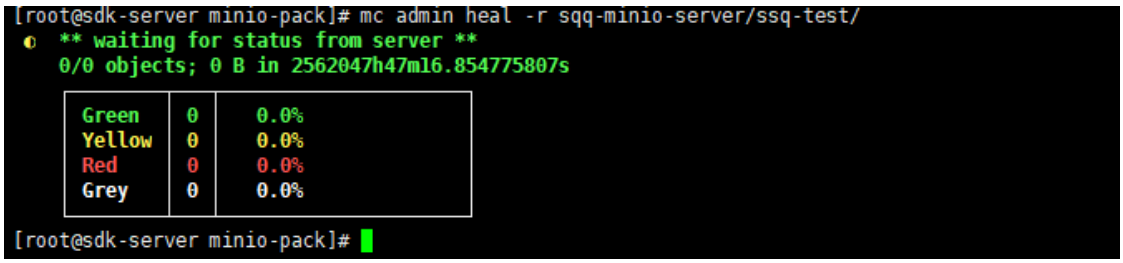
重启服务后即可使用MinIO存储文件



四、数据恢复

当单节点故障，或者单块盘故障恢复后，需要恢复MinIO数据均衡和数据冗余SLA

```
mc admin heal -r ssq-minio/ssq-test   #递归恢复存储桶和对象
```



五、存储扩容

垂直扩容： 直接采用垂直扩容方式扩容MinIO集群的节点磁盘空间，会为集群运行带来若干问题，官方也并不推荐。

水平扩容：

- 对等扩容：例如原集群包含2个节点2块磁盘，则在扩容时必须同样增加2个节点2块磁盘（或为其倍数），以便系统维持相同的数据冗余SLA，从而极大地降低扩容的复杂性；在扩容后，MinIO集群并不会对全部的4个节点进行完全的数据均衡，而是将原本的2个节点视作一个区域，新加入的2节点视作另一区域；当有新对象上传时，集群将依据各区域的可用空间比例确定存放区域，在各区域内仍旧通过哈希算法确定对应的纠删组进行最终的存放。

对等扩容

Hostname	IP	数据盘	备注
sdk-server	192.168.10.100	/	混合云服务、Nginx
minio-node-01	192.168.10.102	/minio_data1、/minio_data2	MioIO原始节点
minio-node-02	192.168.10.103	/minio_data1、/minio_data2	MioIO原始节点
minio-node-03	192.168.10.104	/minio_data1、/minio_data2	MioIO扩容节点
minio-node-04	192.168.10.105	/minio_data1、/minio_data2	MioIO扩容节点

```
# cat >> /data/minio/run/run.sh <<"EOF"
#!/bin/bash
export MINIO_ROOT_USER=ssq
export MINIO_ROOT_PASSWORD=Bestsign@2019

/data/minio/run/minio server  --config-dir /etc/minio --address ":9000" --
console-address ":9001" \
http://192.168.10.10{2...3}/minio_data{1...2} \
http://192.168.10.10{4...5}/minio_data{1...2} >
/data/minio/log/minio_server.log

EOF
```

之后重启每台节点上的MinIO服务

六、存储迁移

6.1、mc客户端

mc客户端迁移
mc mirror 源对象存储/指定桶 目标对象存储/目标桶

```
# mc mirror  sqq-minio-server/ssq-test minio-new-server/ssq-test
```

6.2、rclone客户端

使用rclone： 开源的对象存储在线迁移工具，用于文件和目录的同步，支持阿里云的oss、minio 、亚马逊S3
下载安装rclone

```
# wget https://downloads.rclone.org/rclone-current-linux-amd64.zip --no-
check-certificate
# unzip rclone-current-linux-amd64.zip
# cd rclone-current-linux-amd64
# cp rclone /usr/sbin
```

rclone配置文件

```
# vim /root/.config/rclone/rclone.conf
[minio-old]                     #声明旧的对象存储
type = s3                       #指定S3
provider = Minio
env_auth = false
access_key_id = xGnTDI8gCpdQGEJaqL1M
# access_key
secret_access_key = MCNNrMsCGls4v1PZdganKDHc6vXD5SLZ9jKh7voP     #secret_key
region = cn-east-1
endpoint = http://192.168.10.100:9000                          #旧对象存储地址
location_constraint =
server_side_encryption =

[minio-new]                     #声明旧的对象存储
type = s3
provider = Minio
env_auth = false
access_key_id = ry8AnrY8GapVbw8z
secret_access_key = ziiramB8RVSmIIIsFA6XTFF751mRQTMh
region = cn-east-1
endpoint = http://101.42.53.31:9000                          #旧对象存储地址
location_constraint =
server_side_encryption =
```

执行迁移命令

rclone sync 源对象存储声明:源Bucket 目标对象存储声明:目标Bucket

```
# rclone sync minio-old:ssq-test minio-new:ssq-test
```