腾讯课堂 零声教育 整理

网页版本：https://www.yuque.com/linuxer/xngi03/gpno6ggxl8g7u6gg?singleDoc#%20%E3%80%8Aceph%E5%8D%95%E6%9C%BAdocker%E6%90%AD%E5%BB%BA%E3%80%8B

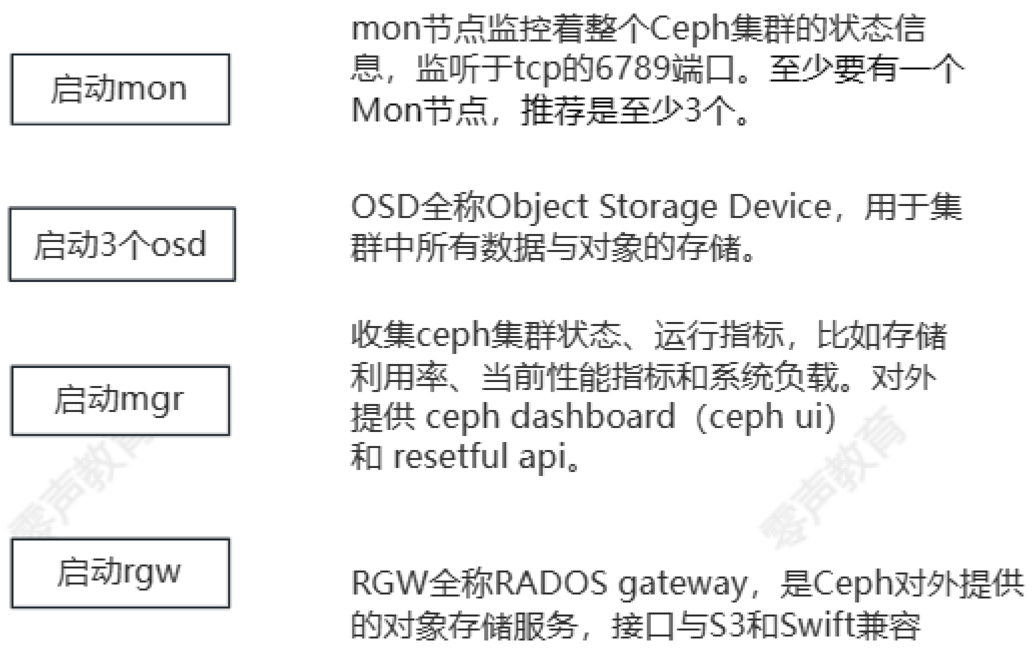
1 安装docker

如果已经安装了不需要重新安装。如果没有安装参考docker课程。

2 部署ceph

部署ceph之前，先学习ceph原理，是否不容易理解不同组件之间的关系。

2.1 部署流程



mon节点监控着整个Ceph集群的状态信息，监听于tcp的6789端口。至少要有一个Mon节点，推荐是至少3个。

OSD全称Object Storage Device，用于集群中所有数据与对象的存储。

收集ceph集群状态、运行指标，比如存储利用率、当前性能指标和系统负载。对外提供 ceph dashboard（ceph ui）和 resetful api。

RGW全称RADOS gateway，是Ceph对外提供的对象存储服务，接口与S3和Swift兼容

要用root用户创建，或有sudo权限。

2.2 创建Ceph专用网络

```
docker network create --driver bridge --subnet 172.20.0.0/16 ceph-network
docker network inspect ceph-network
```

显示

```
[
{
"Name": "ceph-network",
"Id": "34478a9c71a7c07685e68d06c7fe0fdfc9ecff7a12ecd8380dcc4e92a26de959",
"Created": "2023-03-06T23:25:54.211322845-08:00",
"Scope": "local",
"Driver": "bridge",
"EnableIPv6": false,
"IPAM": {
"Driver": "default",
"Options": {},
"Config": [
{
"Subnet": "172.20.0.0/16"
}
]
},
"Internal": false,
"Attachable": false,
"Ingress": false,
"ConfigFrom": {
"Network": ""
},
"ConfigOnly": false,
"Containers": {},
"Options": {},
"Labels": {}
}
]
```

2.3 删除旧的ceph相关容器

删除旧的容器

```
docker rm -f $(docker ps -a | grep ceph | awk '{print $1}')
```

清理旧的ceph相关目录文件，假如有的话

```
rm -rf /etc/ceph /var/lib/ceph /var/log/ceph
```

创建相关目录及修改权限，用于挂载volume

```
mkdir -p /etc/ceph   /var/lib/ceph  /var/lib/ceph/osd /var/log/ceph
```

授权

```
chown -R 64045:64045 /var/lib/ceph/osd
chown -R 64045:64045 /etc/ceph
chown -R 64045:64045 /var/log/ceph
```

2.4 拉取ceph镜像

目前最新的版本是quincy，我们这里使用luminous版本。官网：
https://docs.ceph.com/en/latest/releases/luminous/

```
docker pull ceph/daemon:latest-luminous
```

2.5 搭建mon节点

```
docker run -d --name ceph-mon -p 6789:6789 --network ceph-network --ip 172.20.0.10 -
e CLUSTER=ceph -e WEIGHT=1.0 -e MON_IP=172.20.0.10 -e MON_NAME=ceph-mon -e
CEPH_PUBLIC_NETWORK=172.20.0.0/16 -v /etc/ceph:/etc/ceph -v
/var/lib/ceph/:/var/lib/ceph/ -v /var/log/ceph/:/var/log/ceph/ ceph/daemon:latest-
luminous mon
```

2.6 搭建osd节点

```
docker exec ceph-mon ceph auth get client.bootstrap-osd -o /var/lib/ceph/bootstrap-
osd/ceph.keyring
```

2.7 修改配置文件以兼容etx4硬盘

sudo vim /etc/ceph/ceph.conf

在文件最后添加:
osd max object name len = 256
osd max object namespace len = 64

2.8 分别启动三个容器来模拟集群

```
docker run -d --privileged=true --name ceph-osd-1 --network ceph-network --ip
172.20.0.11 -e CLUSTER=ceph -e WEIGHT=1.0 -e MON_NAME=ceph-mon -e MON_IP=172.20.0.10
-e OSD_TYPE=directory -v /etc/ceph:/etc/ceph -v /var/lib/ceph/:/var/lib/ceph/ -v
/var/lib/ceph/osd/1:/var/lib/ceph/osd -v /etc/localtime:/etc/localtime:ro
ceph/daemon:latest-luminous osd

docker run -d --privileged=true --name ceph-osd-2 --network ceph-network --ip
172.20.0.12 -e CLUSTER=ceph -e WEIGHT=1.0 -e MON_NAME=ceph-mon -e MON_IP=172.20.0.10
-e OSD_TYPE=directory -v /etc/ceph:/etc/ceph -v /var/lib/ceph/:/var/lib/ceph/ -v
/var/lib/ceph/osd/2:/var/lib/ceph/osd -v /etc/localtime:/etc/localtime:ro
ceph/daemon:latest-luminous osd

docker run -d --privileged=true --name ceph-osd-3 --network ceph-network --ip
172.20.0.13 -e CLUSTER=ceph -e WEIGHT=1.0 -e MON_NAME=ceph-mon -e MON_IP=172.20.0.10
-e OSD_TYPE=directory -v /etc/ceph:/etc/ceph -v /var/lib/ceph/:/var/lib/ceph/ -v
/var/lib/ceph/osd/3:/var/lib/ceph/osd -v /etc/localtime:/etc/localtime:ro
ceph/daemon:latest-luminous osd
```

2.9 搭建mgr节点

```
docker run -d --privileged=true --name ceph-mgr --network ceph-network --ip
172.20.0.14 -e CLUSTER=ceph -p 7000:7000 --pid=container:ceph-mon -v
/etc/ceph:/etc/ceph -v /var/lib/ceph/:/var/lib/ceph/ ceph/daemon:latest-luminous mgr
```

2.9.1 开启管理界面

```
docker exec ceph-mgr ceph mgr module enable dashboard
```

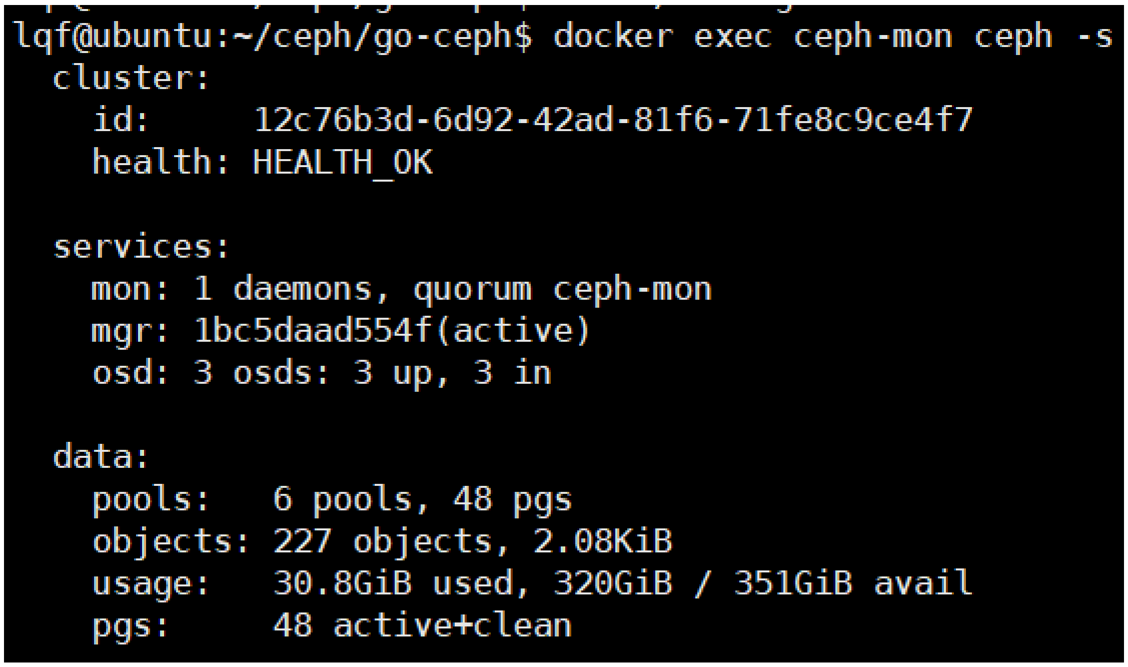
2.10 搭建rgw节点

```
docker exec ceph-mon ceph auth get client.bootstrap-rgw -o /var/lib/ceph/bootstrap-
rgw/ceph.keyring

docker run -d --privileged=true --name ceph-rgw --network ceph-network --ip
172.20.0.15 -e CLUSTER=ceph -e RGW_NAME=ceph-rgw -p 7480:7480 -v
/var/lib/ceph/:/var/lib/ceph/ -v /etc/ceph:/etc/ceph -v
/etc/localtime:/etc/localtime:ro ceph/daemon:latest-luminous rgw
```

2.11 检查Ceph集群状态

```
docker exec ceph-mon ceph -s
```



```
lqf@ubuntu:~/ceph/go-ceph$ docker exec ceph-mon ceph -s
  cluster:
    id:     12c76b3d-6d92-42ad-81f6-71fe8c9ce4f7
    health: HEALTH_OK

  services:
    mon: 1 daemons, quorum ceph-mon
    mgr: 1bc5daad554f(active)
    osd: 3 osds: 3 up, 3 in

  data:
    pools:   6 pools, 48 pgs
    objects: 227 objects, 2.08KiB
    usage:   30.8GiB used, 320GiB / 351GiB avail
    pgs:     48 active+clean
```

2.12 测试添加rgw用户

添加账号

```
docker exec ceph-rgw radosgw-admin user create --uid="testuser" --display-name="lqf
User"
```

下面这些信息在使用s3cmd工具和程序上传下载的时候需要使用。
{
"user": "testuser",
"access_key": "YZLDI5I2CEM4PEENTX9V",
"secret_key": "OzsQSO7jJJveMJSObuEMd0bqQi1TL1dtPQnWhx5T"
}

查询账号信息

后续使用s3cmd、程序访问ceph时需要用到user对应的access_key和secret_key

```
docker exec ceph-rgw radosgw-admin user info --uid=testuser
```

2.13 客户端测试

1 安装s3cmd

客户机执行 sudo apt install s3cmd

2 配置s3cmd

配置分两个步骤：

1. 执行 s3cmd --configure
2. 修改配置文件/root/.s3cfg，（**要注意的是如果不是root的权限**，这个文件在home目录，比如/home/lqf/.s3cfg）

2.1 s3cmd --configure

客户机执行 s3cmd --configure，开始配置，根据提示输入accessKey,securityKey 生成基本的配置文件

```
lqf@ubuntu:~/ceph/go-ceph$ s3cmd --configure

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

Access key and Secret key are your identifiers for Amazon S3. Leave them empty for
using the env variables.
Access Key: D3HTA2TRXBBE514USEQT      ##此处填上一步获取到的access_key
Secret Key: AZxoYU6u3DkLUw9OMnRewfx73DxhjpDICwSjEIwH      #此处填上一步获取到的
secret_key
Default Region [US]:    #默认即可，直接回车

Use "s3.amazonaws.com" for S3 Endpoint and not modify it to the target Amazon S3.
```

```
S3 Endpoint [s3.amazonaws.com]:    #默认即可，直接回车

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

Encryption password is used to protect your files from reading
by unauthorized persons while in transfer to S3
Encryption password:
Path to GPG program [/usr/bin/gpg]:    #默认即可，直接回车

When using secure HTTPS protocol all communication with Amazon S3
servers is protected from 3rd party eavesdropping. This method is
slower than plain HTTP, and can only be proxied with Python 2.7 or newer
Use HTTPS protocol [Yes]: no     #不使用https，填写no

On some networks all internet access must go through a HTTP proxy.
Try setting it here if you can't connect to S3 directly
HTTP Proxy server name:      #默认即可，直接回车

New settings:
Access Key: D3HTA2TRXBBE514USEQT
Secret Key: AZxoYU6u3DkLUw9OMnRewfx73DxhjpDICwSjEIwH
Default Region: US
S3 Endpoint: s3.amazonaws.com
DNS-style bucket+hostname:port template for accessing a bucket: %
(bucket)s.s3.amazonaws.com
Encryption password:
Path to GPG program: /usr/bin/gpg
Use HTTPS protocol: False
HTTP Proxy server name:
HTTP Proxy server port: 0

Test access with supplied credentials? [Y/n] n    #测试访问，此时还没配置完，不测试，填写no

Save settings? [y/N] y    #是否保存，是，填写y
Configuration saved to '/root/.s3cfg'     （具体文件路径根据实际显示）
```

2.2 修改/root/.s3cfg配置文件

还没结束，还要修改刚生成的/root/.s3cfg中的三处配置

cloudfront_host = [serverIP]（改成自己的服务端的IP）
host_base = [serverIP]:[Port]（改成自己的服务端的IP和端口）
host_bucket = [serverIP]:[Port]/%(bucket)（改成自己的服务端的IP和端口）

示例：我服务器本地的ceph集群环境，rgw默认端口为7480， ip填写自己服务器的ip即可
cloudfront_host = 192.168.1.27
host_base = 192.168.1.27:7480
host_bucket = %(bucket)192.168.1.27:7480

3 测试s3cmd命令

创建名为test-bucket的bucket

```
lqf@ubuntu:~/ceph/go-ceph$ s3cmd mb s3://test-bucket
Bucket 's3://test-bucket/' created
```

查看bucket桶列表

```
lqf@ubuntu:~/ceph/go-ceph$ s3cmd ls
2023-03-07 07:52  s3://test-bucket
```

即s3配置正常，可正常连接集群

4 s3cmd常用命令

1. 针对bucket桶的操作：

> **创建bucket**
> ```
> $ s3cmd mb s3://{bucket_name}
> ```
>
> **删除bucket**（bucket需为空）
> ```
> $ s3cmd rb s3://{bucket_name}
> ```
>
> **查看bucket列表或bucket内文件列表**
> ```
> s3cmd ls
> s3cmd ls s3://{bucket_name}
> ```

2. **针对bucket中文件的操作：**

> **上传文件到bucket中**
> ```
> $ s3cmd put fio-fio-3.10.zip s3://test-bucket
> ```
>
> **删除文件**
> ```
> s3cmd del s3://test-bucket/file.txt
> ```
>
> **批量删除文件**
> ```
> s3cmd del s3://test-bucket/aa*
> s3cmd del s3://test-bucket/test/*
> ```
>
> **批量上传文件**
> ```
> $ s3cmd put test/* s3://test-bucket
> ```
>
> **递归上传文件**（可上传整个文件夹-包含文件夹）
> ```
> #-r  递归参数，全称为：--recursive
> $ s3cmd put -r /root/test s3://test-bucket
> ```
>
> **同步目录下文件至bucket中**（应该类似于git合流代码）
> ```
> s3cmd sync ./test/ s3://test-bucket
> ```
>
> **复制bucket中文件到其他bucket中**
> ```
> s3cmd cp s3://test-bucket/aaaa s3://test-bucket-2
> ```
>
> **下载文件**
> ```
> s3cmd get s3://test-bucket/file.txt
> s3cmd get s3://test-bucket/file.txt /root/test/
> ```

3. **针对权限的操作：**

> **将文件权限设置为所有人可读**
> $ s3cmd setacl --acl-public s3://test-bucket/file.txt
>
> **将bucket中整个文件夹设置权限为私有读**（递归权限，文件夹下所有文件都生效）
> $ s3cmd setacl --acl-private -r s3://test-bucket/test/

4. **更多参考见官方**

https://tecadmin.net/install-s3cmd-manage-amazon-s3-buckets/

3 go ceph客户端操作

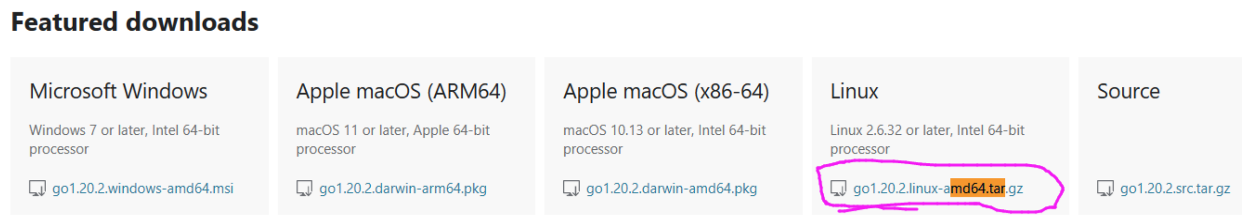
我们使用go 第三方库 AWS公司的API进行操作，讲解代码之前先介绍AWS S3相关术语：

- Region：存储数据所在的地理区域区域；
- Endpoint：存储服务入口,Web服务入口点的URL；
- Bucket：存储桶是S3中用于存储对象的容器
- Object：对象是S3中存储的基本实体,由对象数据和元数据组成对象是S3中存储的基本实体，由对象数据和元数据组成；
- Key：键是存储桶中对象的唯一标识符，桶内的每个对象都只能有一个key。

3.1 安装go工具链

3.1.1 下载go安装包

olang官网下载地址： https://golang.google.cn/dl/


Featured downloads

Microsoft Windows
Windows 7 or later, Intel 64-bit processor
go1.20.2.windows-amd64.msi

Apple macOS (ARM64)
macOS 11 or later, Apple 64-bit processor
go1.20.2.darwin-arm64.pkg

Apple macOS (x86-64)
macOS 10.13 or later, Intel 64-bit processor
go1.20.2.darwin-amd64.pkg

Linux
Linux 2.6.32 or later, Intel 64-bit processor
go1.20.2.linux-amd64.tar.gz

Source
go1.20.2.src.tar.gz

注意系统和版本的区别

```
wget https://dl.google.com/go/go1.20.2.linux-amd64.tar.gz
```

3.1.2 解压和设置环境变量

将其解压缩到/usr/local/(会在/usr/local中创建一个go目录)

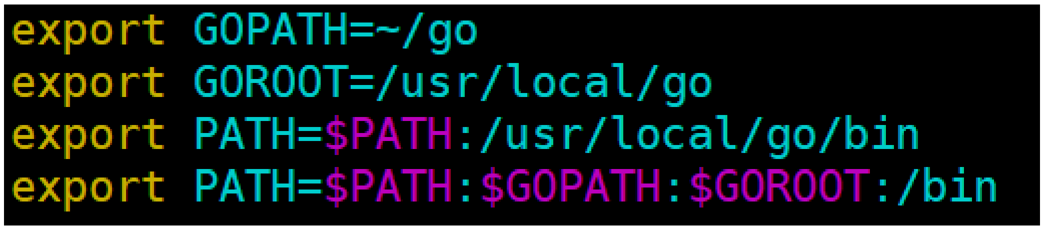
```
sudo tar -C /usr/local -xzf go1.20.2.linux-amd64.tar.gz
```

添加环境变量

```
sudo vim /etc/profile
```

在打开的文件最后添加:

```
#在home目录下的go目录GOPATH
export GOPATH=~/go
export GOROOT=/usr/local/go
export PATH=$PATH:/usr/local/go/bin
export PATH=$PATH:$GOPATH:$GOROOT:/bin
```



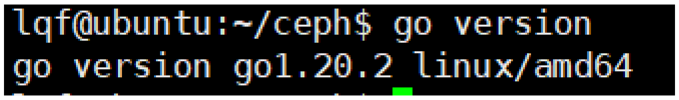
```
export GOPATH=~/go
export GOROOT=/usr/local/go
export PATH=$PATH:/usr/local/go/bin
export PATH=$PATH:$GOPATH:$GOROOT:/bin
```

wq保存退出后source一下

```
source /etc/profile
```

查看版本

```
go version
```



```
lqf@ubuntu:~/ceph$ go version
go version go1.20.2 linux/amd64
```

测试使用

在你的工作区创建hello.go

```
package main
import "fmt"
func main() {
    fmt.Printf("hello, world\n")
}
```

构建项目（Then build it with the `go` tool)
go build hello.go
会生成一个名为hello的可执行文件
执行项目

$ **./hello**
hello, world

3.1.3 设置代理

go提供了带量的第三方库，单不少都是国外的源导致国内下载速度非常慢，所以需要设置代理

设置代理

```
go env -w GOPROXY=https://goproxy.cn,direct
```

3.2 测试代码和编译运行

3.2.1 创建工程和编写代码

// 创建目录
mkdir go-ceph
cd go-ceph
//初始化工程，格式 go mod init 工程名， 生成的执行文件即是工程名
go mod init go-ceph

编写一个main.go实现文件上传下载

```go
package main

import (
    "fmt"
    "io/ioutil"
    "time"

    "gopkg.in/amz.v1/aws"
    "gopkg.in/amz.v1/s3"
)

var cephConn *s3.S3 //提供类似s3的接口
// 填写自己的Key
var accessKey = "YZLDI5I2CEM4PEENTX9V"
var secretKey = "OzsQSO7jJJveMJSObuEMdObqQi1TL1dtPQnWhx5T"

// 填写自己服务器的地址
var url = "http://192.168.1.27:7480"

func init() {
    fmt.Println("init app")
    auth := aws.Auth{
        AccessKey: accessKey,
        SecretKey: secretKey,
    }
    region := aws.Region{
        Name:                 "default",
        EC2Endpoint:          url, // "http://<ceph-rgw ip>:<ceph-rgw port>"
        S3Endpoint:           url,
        S3BucketEndpoint:     "",    // Not needed by AWS S3
        S3LocationConstraint: false, // true if this region requires a
LocationConstraint declaration
        S3LowercaseBucket:    false, // true if the region requires bucket names to
be lower case
        Sign:                 aws.SignV2,
    }

    cephConn = s3.New(auth, region)
}

// 获取一个桶
func GetCephBucket(bucket string) *s3.Bucket {
    fmt.Println("get bucket:", bucket)
    return cephConn.Bucket(bucket)
}

// 将本地文件上传到ceph的一个bucket中
func put2Bucket(bucket *s3.Bucket, localPath, cephPath string) (*s3.Bucket, error) {
    fmt.Println("put ", localPath, " to ", cephPath)
    err := bucket.PutBucket(s3.PublicRead)
    if err != nil {
        fmt.Println(err.Error())
        return nil, err
    }

    bytes, err := ioutil.ReadFile(localPath)
    if err != nil {
        fmt.Println(err.Error())
        return nil, err
    }
    // https://developer.mozilla.org/en-
US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types
    err = bucket.Put(cephPath, bytes, "text/plain", s3.PublicRead) // "octet-stream"
    return bucket, err
}

// 从ceph下载文件到本地
func downloadFromCeph(bucket *s3.Bucket, localPath, cephPath string) error {
    fmt.Println("download ", cephPath, " to ", localPath)
    data, err := bucket.Get(cephPath)
    if err != nil {
        fmt.Println(err.Error())
        return err
    }
    return ioutil.WriteFile(localPath, data, 0666)
}
```

```go
// 删除指定的文件
func delCephData(bucket *s3.Bucket, cephPath string) error {
    fmt.Println("delete file: ", cephPath)
    err := bucket.Del(cephPath)
    if err != nil {
        fmt.Println(err.Error())
    }
    return err
}

// 删除桶,删除桶时要保证桶内文件已经被删除
func delBucket(bucket *s3.Bucket) error {
    fmt.Println("delete bucket: ", bucket.Name)
    err := bucket.DelBucket()
    if err != nil {
        fmt.Println("delete bucket failed: ", err.Error())
    }
    return err
}

// 批量获取文件信息
func getBatchFromCeph(bucket *s3.Bucket, prefixCephPath string) []string {
    fmt.Println("bucket List: ", prefixCephPath)
    maxBatch := 100

    // bucket.List() 返回桶内objects的信息，默认1000条
    resultListResp, err := bucket.List(prefixCephPath, "", "", maxBatch)
    if err != nil {
        fmt.Println(err.Error())
        return nil
    }

    keyList := make([]string, 0)
    for _, key := range resultListResp.Contents {
        keyList = append(keyList, key.Key)
    }

    return keyList
}

func main() {
    fmt.Println("hello, ceph")
    bucketName := "bucket_test"
    filename := "go.sum"
    cephPath := "/static/default/bucket_test/V1/" + "go.sum"
    cephPath2 := "/static/default/bucket_test/V1/" + "go2.sum"
    // 获取指定桶
    bucket := GetCephBucket(bucketName)

    // 上传，需要指定bucket，以及存储路径（RESTful风格的路径）
    bucket, err := put2Bucket(bucket, filename, cephPath)
    if err != nil {
        return
    }

    // 上传，需要指定bucket，以及存储路径（RESTful风格的路径）
    bucket, err = put2Bucket(bucket, filename, cephPath2)
    if err != nil {
        return
    }

    // 下载
    localPath := "go-bk.sum"
    err = downloadFromCeph(bucket, localPath, cephPath)
    if err != nil {
        return
    }

    // 获得url
    url := bucket.SignedURL(cephPath, time.Now().Add(time.Hour))
    fmt.Println("SignedURL:", url)

    // 批量查找
    prefixCephpath := "static/default/bucket_test/V1"
    lists := getBatchFromCeph(bucket, prefixCephpath)
    for _, list := range lists {
        fmt.Println(list)
    }

    // 删除数据
    delCephData(bucket, cephPath)

    // 删除桶
    delBucket(bucket)
}
```

3.2.2 编译代码和运行

//设置代理和下载第三方库
go env -w GOPROXY=https://goproxy.cn,direct
go get gopkg.in/amz.v1/aws
go get gopkg.in/amz.v1/s3

// 编译，编译后得到go-ceph
go build

//运行，因为故意没有删除完bucket的文件就删除bucket，所以在删除bucket的时候报错。
./go-ceph

运行打印示例：

> init app
> hello, ceph
> get bucket: bucket_test
> put go.sum to /static/default/bucket_test/V1/go.sum
> put go.sum to /static/default/bucket_test/V1/go2.sum
> download /static/default/bucket_test/V1/go.sum to go-bk.sum
> SignedURL: http://192.168.1.27:7480/bucket_test/static/default/bucket_test/V1/go.sum?AWSAccessKeyId=YZLDI5I2CEM4PEENTX9V&Expires=1678338112&Signature=p1f5zgCLb%2Fn0um%2FJ1EtLUUcad%2F4%3D
> bucket List: static/default/bucket_test/V1
> static/default/bucket_test/V1/go.sum
> static/default/bucket_test/V1/go2.sum
> delete file: /static/default/bucket_test/V1/go.sum
> delete bucket: bucket_test
> delete bucket failed: 409 Conflict

代码打印的SignedURL: http://192.168.1.27:7480/bucket_test2/static/default/bucket_test/V1/go.sum?AWSAccessKeyId=YZLDI5I2CEM4PEENTX9V&Expires=1678375143&Signature=MlM98%2FIQsFgOnTbNrnNFCSeYLt0%3D

- AWSAccessKeyId=YZLDI5I2CEM4PEENTX9V 是用户的access_key
- Expires=1678375143是过期的时间，比如1678375143是2023-03-09 23:19:03，过了这个时间就不能再访问，需要重新申请权限
- Signature=MlM98%2FIQsFgOnTbNrnNFCSeYLt0%3D 是通过secret_key+YZLDI5I2CEM4PEENTX9V签名的key服务器需要对该key做匹配。

这个url为什么不能访问？实际是因为代码将对应的文件删除了，**所以如果需要在浏览器访问则需要将末尾的删除文件delCephData、删除桶delBucket注释**

```
    // 删除数据
    //delCephData(bucket, cephPath)

    // 删除桶
    //delBucket(bucket)
```

然后再运行获取新URL就可以在浏览器访问了。

更多文档：https://gopkg.in/amz.v1/aws

参考

- 使用s3cmd访问ceph的对象存储服务

- ceph 分布式存储系统
- 第一篇：用Docker搭建Ceph集群(nautilus版本):
- 使用 docker 快速部署 ceph - 腾讯云开发者社区-腾讯云 (tencent.com)
- ceph对象存储具体含义 - 对象存储oss - 新睿云 (xinruiyun.cn)