目录

[一、 系统搭建 2](#_Toc91790150)

[实验环境 2](#_Toc91790151)

[服务器搭建：Minio 2](#_Toc91790152)

[客户端安装：Minio Client 3](#_Toc91790153)

[二、 性能观测 4](#_Toc91790154)

[安装s3bench工具 4](#_Toc91790155)

[使用s3bench测试 5](#_Toc91790156)

[三、 尾延迟挑战 8](#_Toc91790157)

[原始尾延迟 8](#_Toc91790158)

[对冲请求Hedged Request 8](#_Toc91790159)

[关联请求Tied Request 9](#_Toc91790160)

# 系统搭建

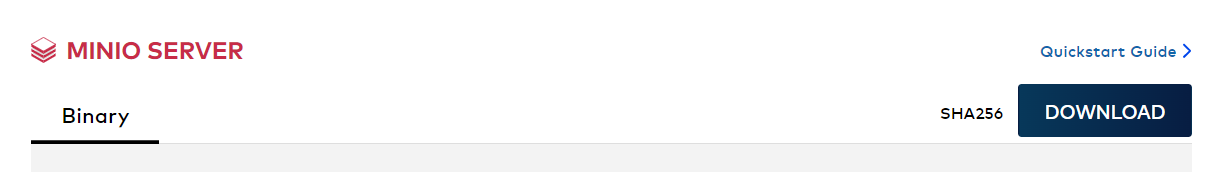
## 实验环境

操作系统：Windows10

## 服务器搭建：Minio

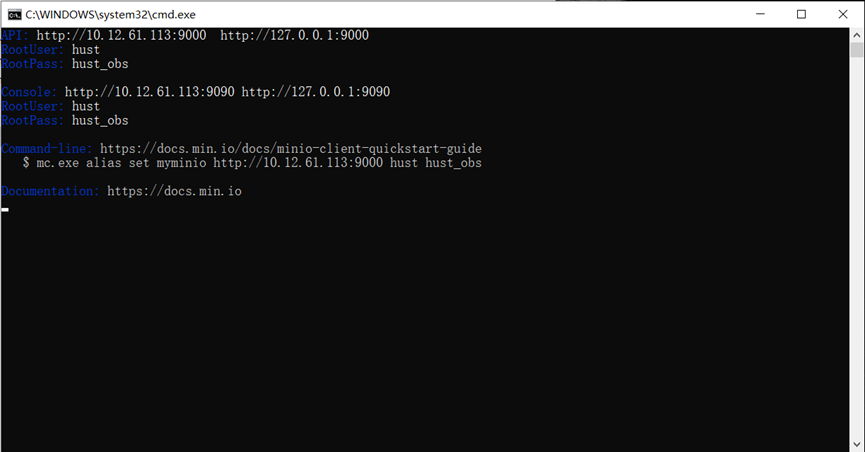
第一步，下载Minio Server

最新版网址： <https://minio.io/downloads.html>，进入网页，点击download。



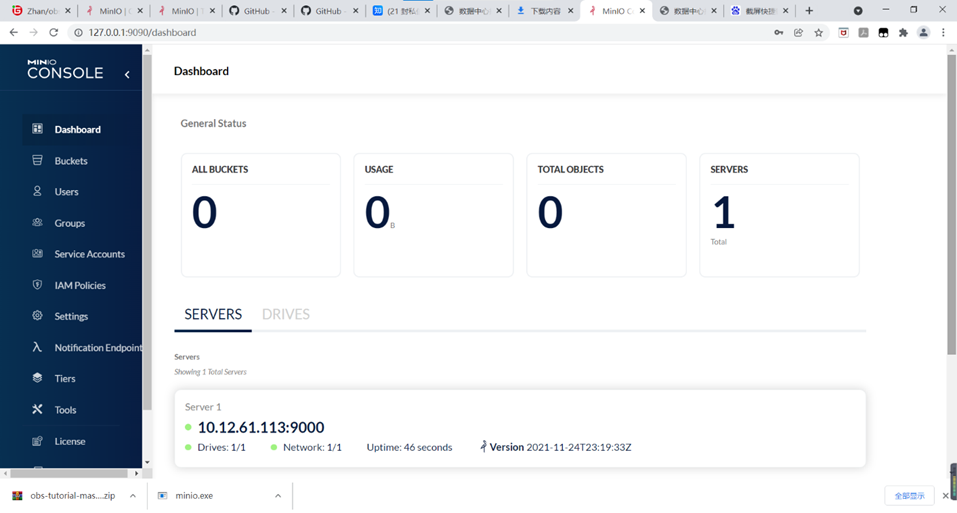
下载minio.exe后，放入实验文件夹。

第二步，执行实验文件夹中的run-minio.sh，配置账号密码。



如图所示，用户名为：hust；用户密码为：hust\_obs。服务器页面端口为：<http://127.0.0.1:9000>。

第三步，登录页面，管理服务端。



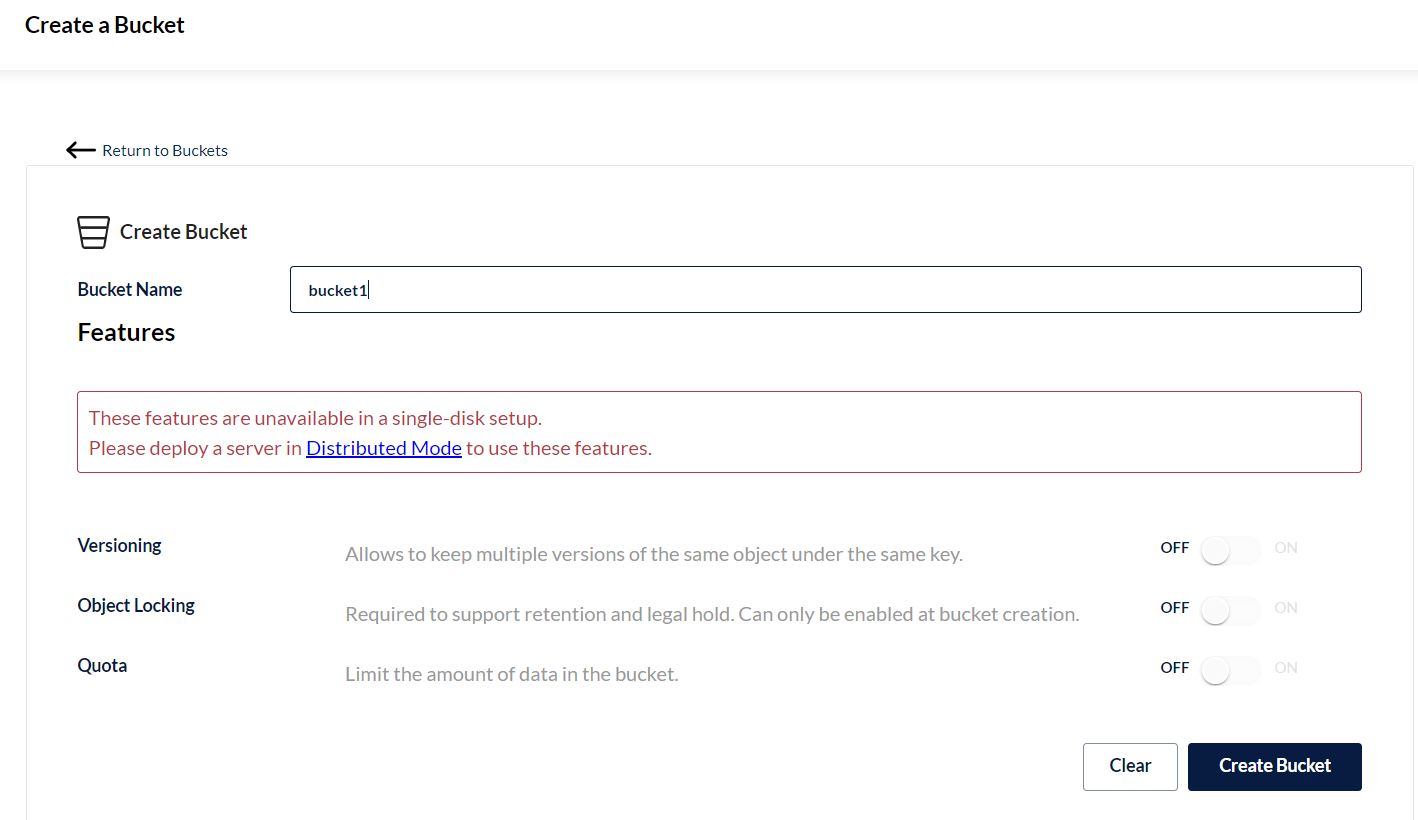
可以看见刚刚配置的1个服务器。

## 客户端安装：Minio Client

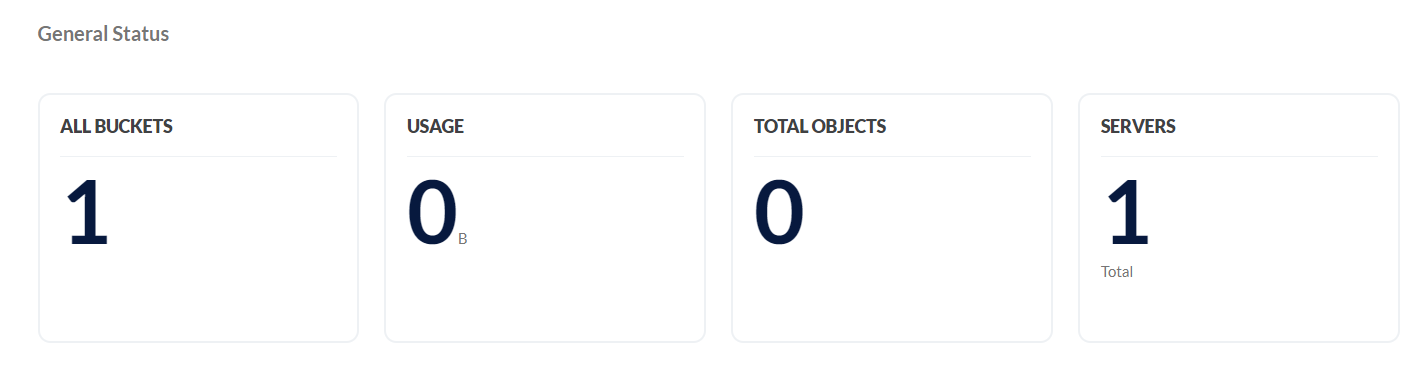
Windows10平台上Minio服务器与客户端是一体的，可在网页上进行bucket的添加。以下是详细步骤：

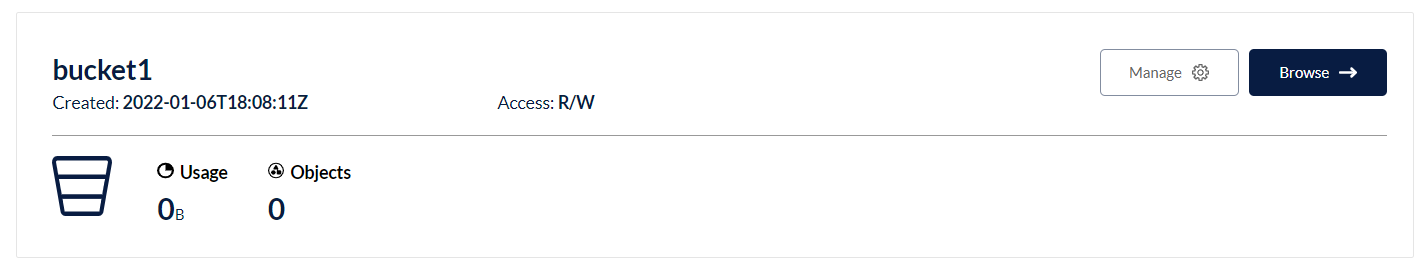
第一步，登录服务器后台。

第二步，添加bucket.

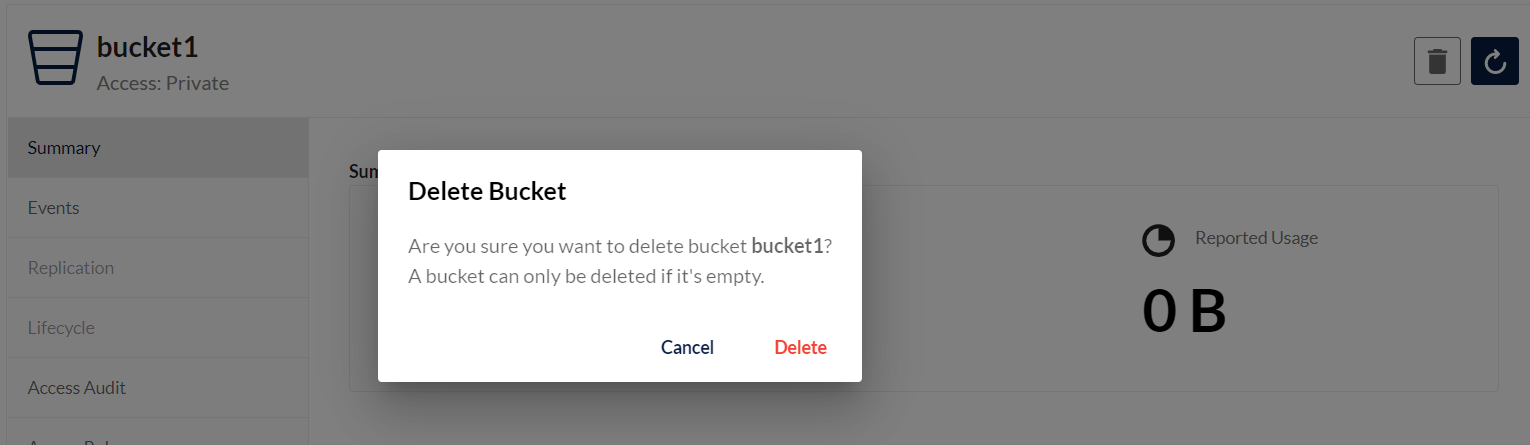


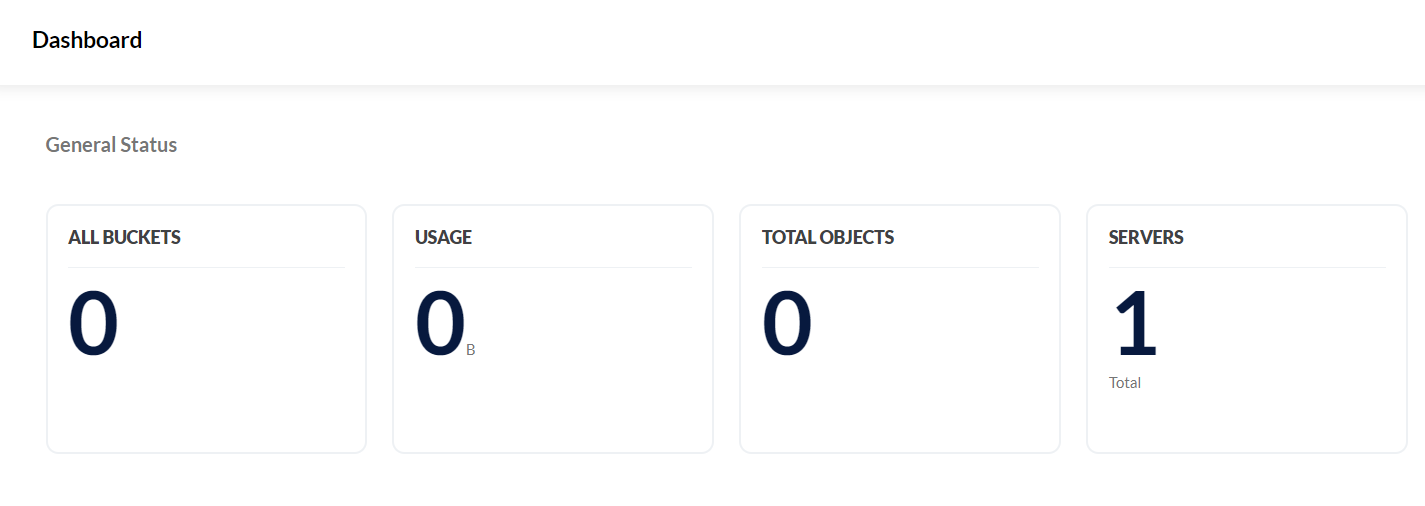
第三步，查看后台，bucket的数量变为1.





第四步，删除bucket1，bucket数量变为0.





# 性能观测

## 安装s3bench工具

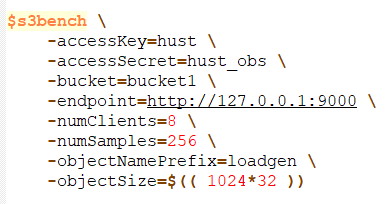
1. 安装go环境：

sudo apt-get install golang

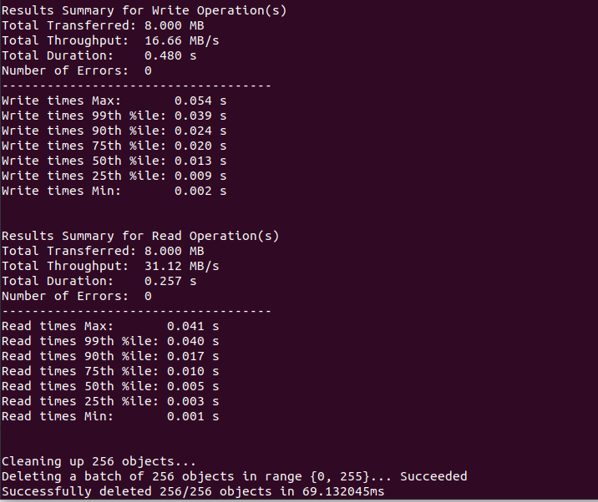
1. 安装s3bench

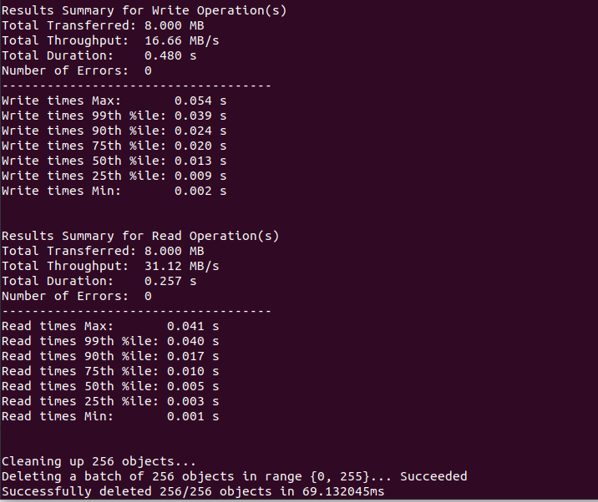
go get -u github.com/igneous-systems/s3bench

1. 修改run-s3bench.sh中的-bucket参数为自己的bucket名：



运行结果：

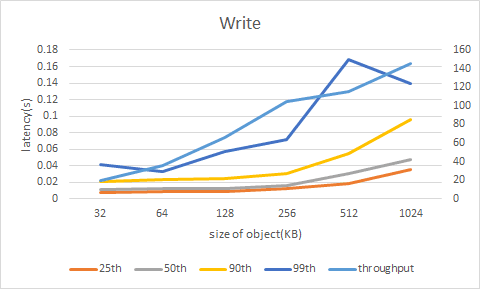


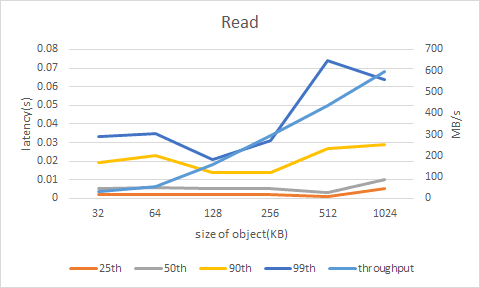


## 使用s3bench测试

1. 对象尺寸对性能的影响

选取对象尺寸分别为：32, 64, 128, 256, 512, 1024（KB），观察吞吐量和尾延迟。





观察可得：

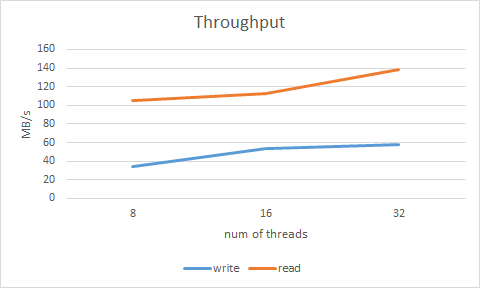
对象尺寸越大，读写吞吐量越大；

对象尺寸越大，写的中位数延迟和尾延迟越大，读操作亦有此趋势，但在尺寸很小（<128KB）时，现象不明显。

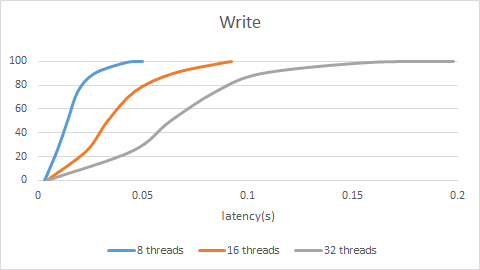
1. 并发数对性能的影响

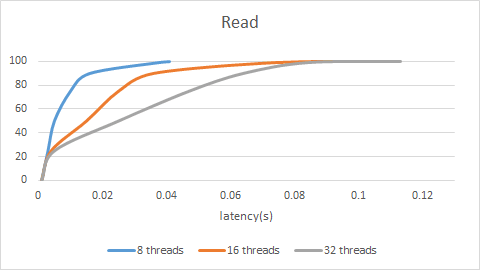
设置object size = 128 KB，分别测试线程数为8, 16, 32时的性能。

吞吐量：



读写尾延迟：



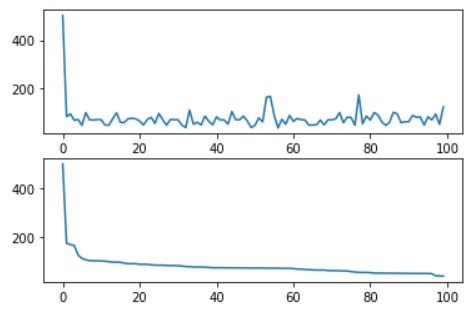


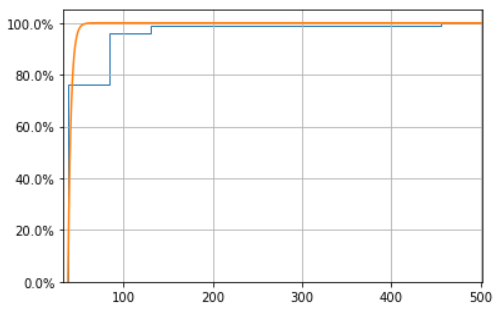
观察可得：

并发数越大，吞吐量越高，尾延迟也越大。

# 尾延迟挑战

## 原始尾延迟

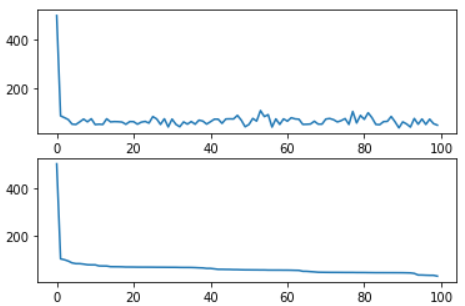


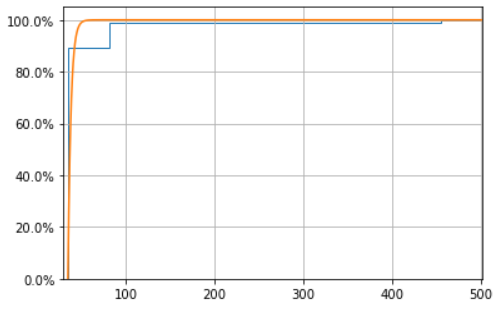


## 关联请求Tied Request

原理：多个服务器同时发出同一个请求，哪个延迟短使用哪个。

实验结果：





由实验结果可知，关联请求尾延迟结果相对原始结果有较大改善，99%从约130ms下降为约75ms