### 实验一:系统搭建

#### 配置Python环境

1. 下载[Minconda3 Windows 64-bit](https://docs.conda.io/en/latest/miniconda.html)
2. 添加环境变量：在高级系统里找到环境变量->系统变量->在path添加condabin路径
3. 创建实验的python环境：conda create --name datacenter python=3.6

关于conda和pip安装包的一些问题：

1）用户环境pip安装，非conda环境下全局使用。

2）conda某个环境下pip安装，仅能在该虚拟环境下使用。

3）用户环境conda安装，在conda共享包目录下存放，base环境可以直接使用，用户环境不能使用，其它虚拟环境安装同样的包时，先在共享包目录下寻找。

4）conda某个环境下conda安装，首先会统一放到一个共享目录，然后复制到该环境的site-packages文件下。

#### 服务端Minio

1. 下载最新版Minio：<https://min.io/download>
2. 将minio.exe移动到服务端的目录
3. 设置Minio环境变量，方便网页客户端的登录

(临时环境变量，当时cmd环境起作用)

set MINIO\_ROOT\_USER=hust

set MINIO\_ROOT\_PASSWORD=hust\_obs

(用户环境变量，当前用户环境起作用)

setx MINIO\_ROOT\_USER hust

setx MINIO\_ROOT\_PASSWORD hust\_obs

(系统环境变量，系统环境起作用，需要管理员权限执行命令)

setx /m MINIO\_ROOT\_USER hust

setx /m MINIO\_ROOT\_PASSWORD hust\_obs

1. 启动server服务

minio -C ./ server ./r­­­­­oot –console-address :9090

(-C参数指定配置文件存储的路径，“./”表示当前路径，server命令启动服务，后面跟着数据存放的目录，所有桶和对象都存储在当前目录的root的目录下)

1. 访问minio客户端，server启动输出的任意一个网页API地址都可以访问

账号和密码在启动信息中显示，也就是之前设置的hust，hust\_obs

### 实验二：性能观测

### 评测工具

选择S3 Bench，主要原因是操作笔记方便，命令简单，执行命令脚本：

s3bench.exe ^

-accessKey=hust ^

-accessSecret=hust\_obs ^

-bucket=loadgen ^

-endpoint=http://127.0.0.1:9000 ^

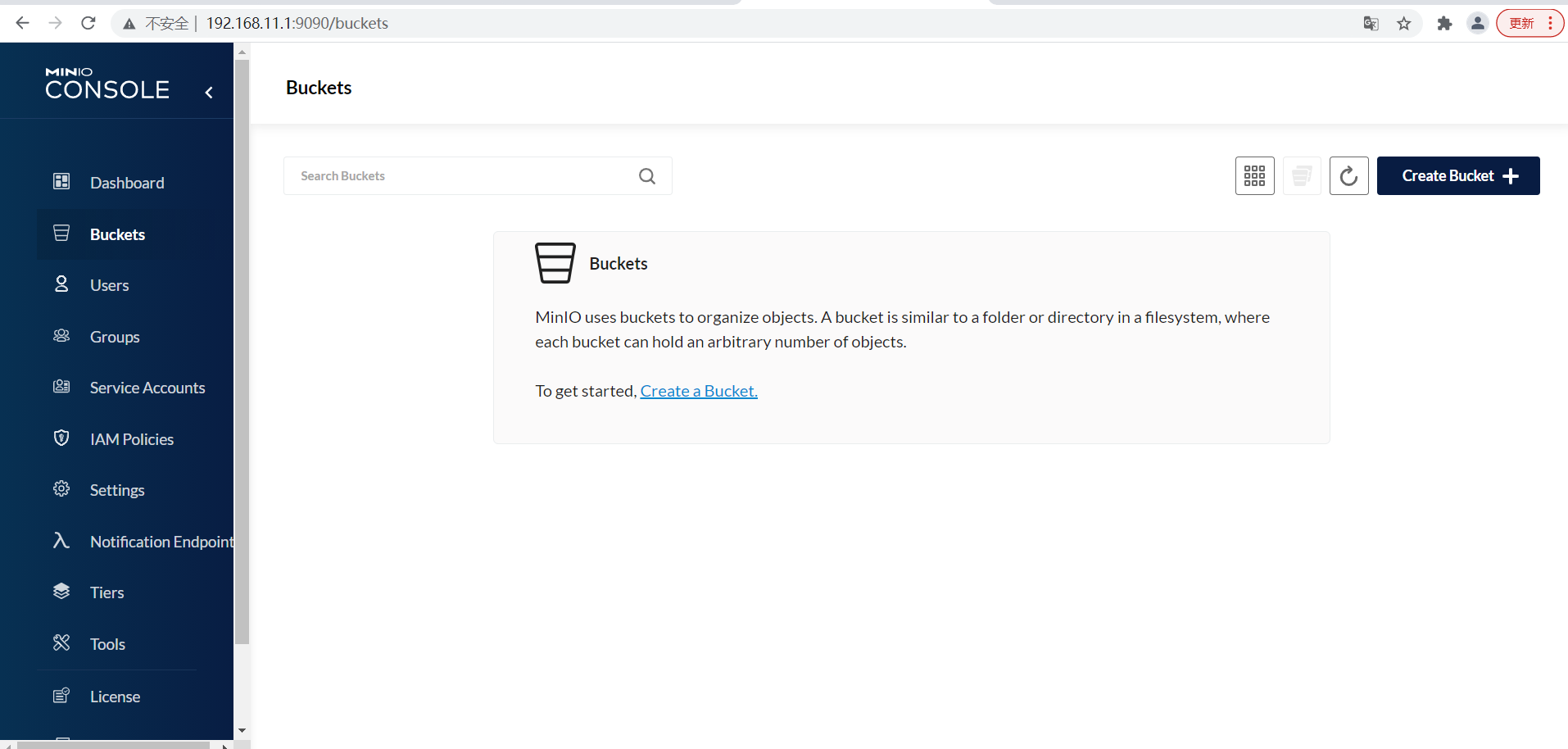
-numClients=8 ^

-numSamples=256 ^

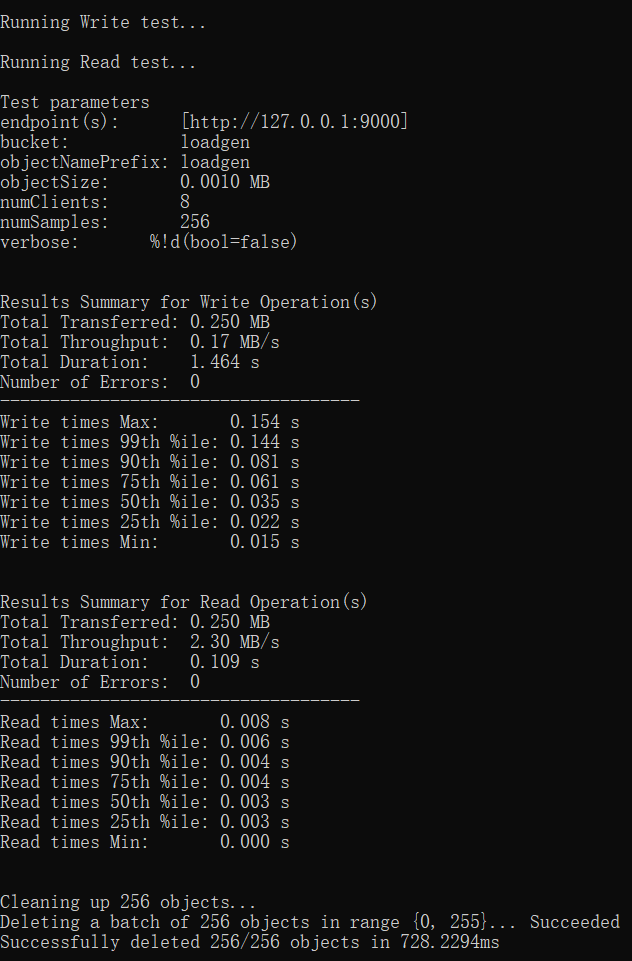
-objectNamePrefix=loadgen ^

-objectSize=1024

在执行脚本之前，要先创建我们的存储桶，不然会全部报错。创建桶的操作通过Minio客户端完成，在网站上访问客户端API，输入用户密码登录，在控制台中Buckets界面创建。要注意我们创建的桶的名称要和脚本中桶的名称保持一致。



创建了桶后，我们就可以运行脚本了。

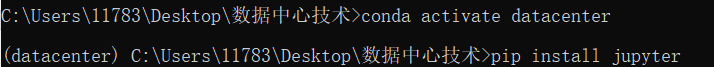


### 实验三：尾延迟挑战

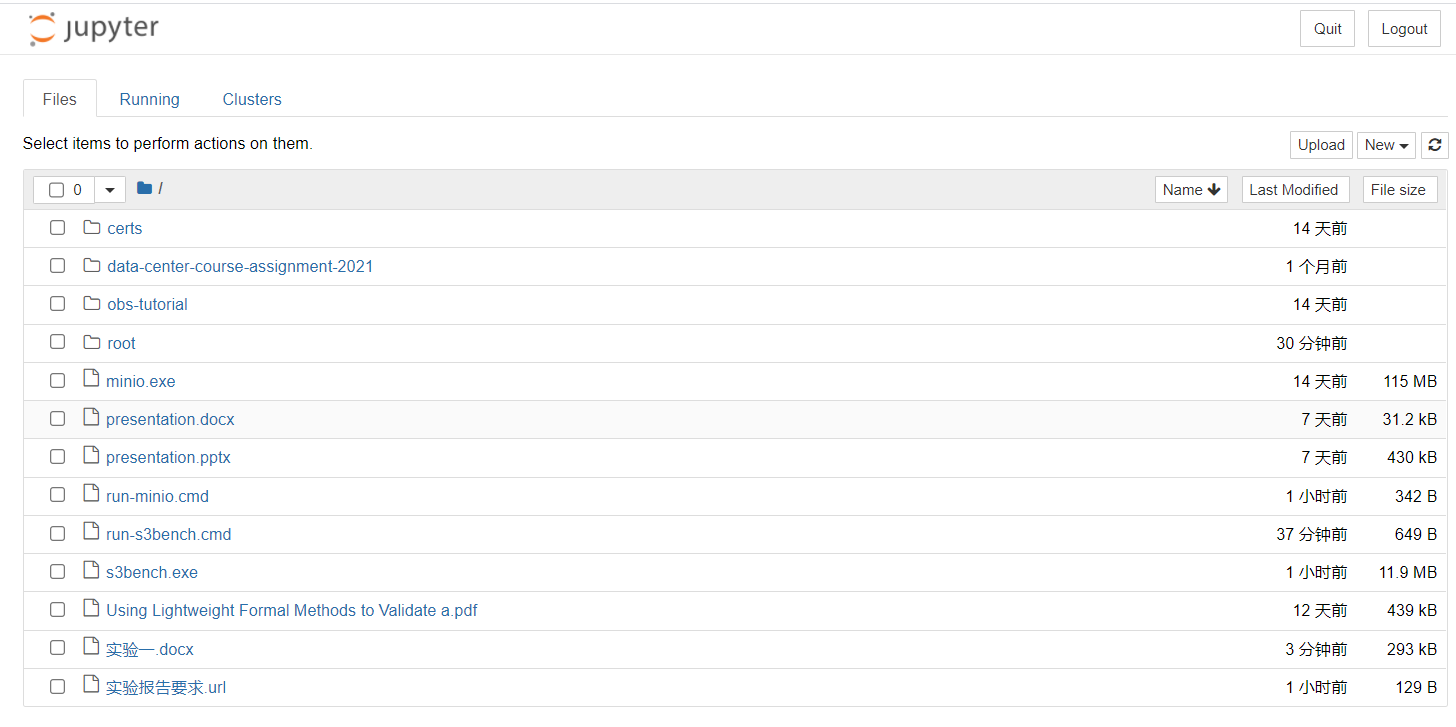
1. 工具介绍

Jupyter Notebook是一个交互式笔记本，支持运行 40 多种编程语言。Jupyter Notebook 的本质是一个 Web 应用程序，便于创建和共享文学化程序文档，支持实时代码，数学方程，可视化和 markdown。它的主要用途是：数据清理和转换，数值模拟，统计建模，机器学习等，Jupyter Notebook 与 IPython终端 共享同一个内核。

我们先进入实验的python环境，再通过pip命令下载Jupyter Notebook应用程序。代码如下：



然后在命令行输入jupyter notebook打开应用程序。



在Jupyter Notebook上创建python文件时，可以选择内核环境，只有一个默认的python3，没有用conda创建的实验环境。为了使用datacenter的python环境，我们还要安装一个nb\_conda插件，然后重启Jupyter Notebook就可以切换到conda环境。但是要注意，nb\_conda和python版本是否冲突，实验证明3.9以上的python环境无法安装。由于我的系统环境是3.10，conda base环境是3.9，都无法安装，所以我只能在conda datacenter(python=3.6)环境下安装nb\_conda，也只有在datacenter环境启动Jupyter Notebook才会显示conda虚拟环境。

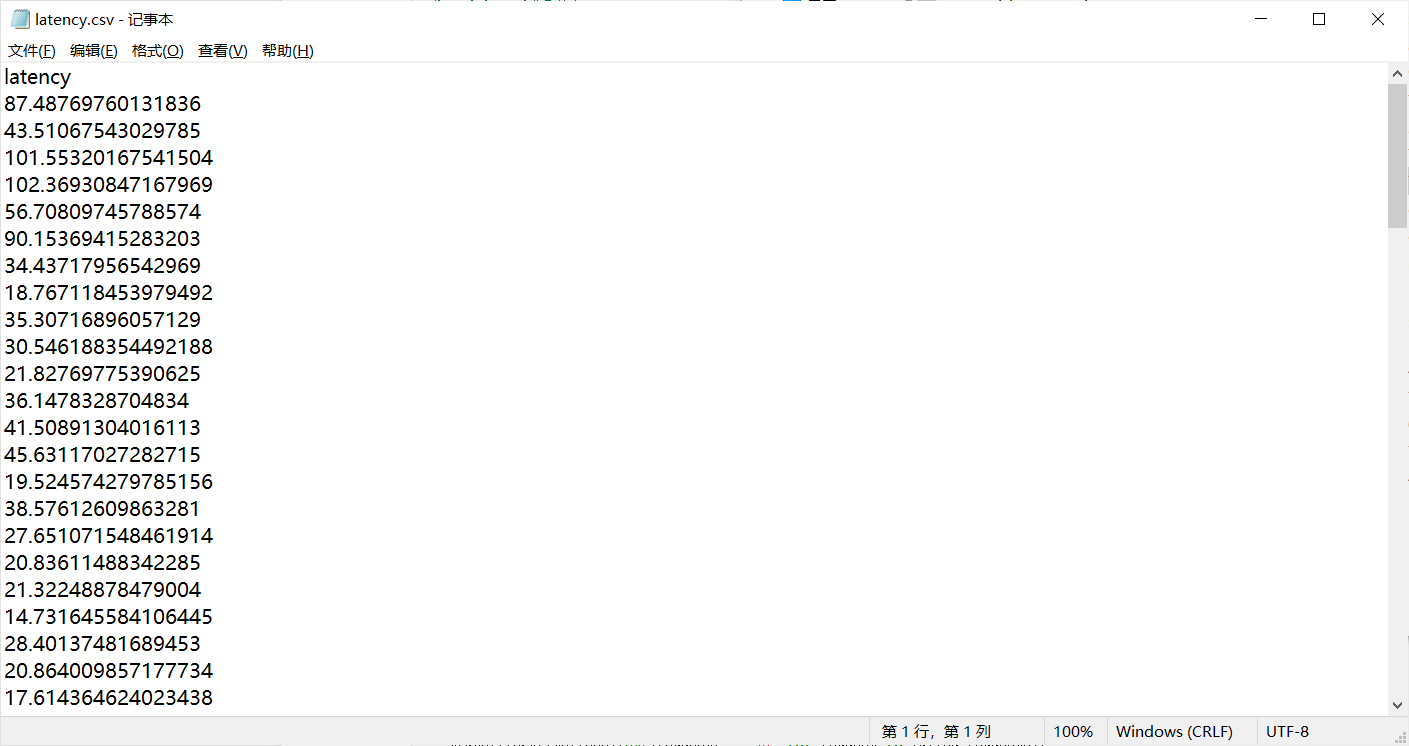
1. 编写程序记录延迟

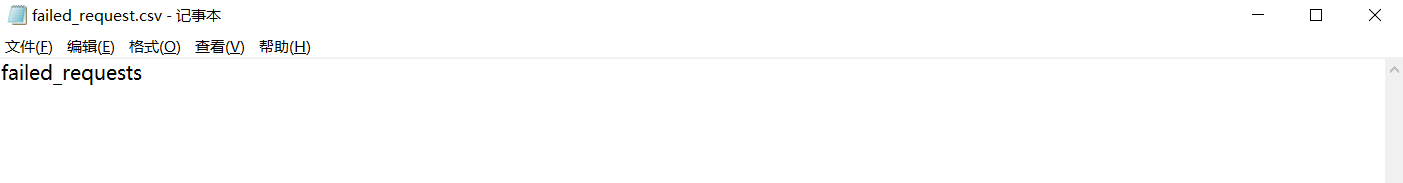




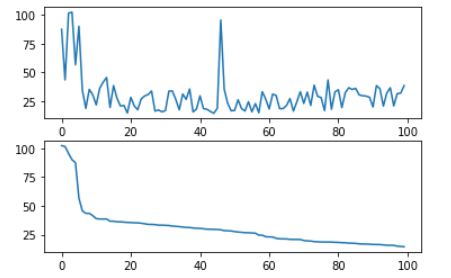


延迟结果和请求出错结果都存在本地的csv文件中，如图所示：

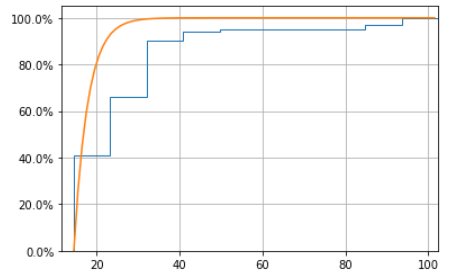




为了更明显地观察尾延迟现象，画出延迟的分布情况：



下面是排队论模型拟合曲线：



1. 尝试减轻尾延迟现象

刚开始打算启动一个新的线程来请求上传文件，写一个循环监测请求时间，如果延迟超过50ms，则再启动一个线程重新发起请求，然而线程启动时间有一百毫秒以上，远远大于程序的最大延迟，所有不能用启动新线程的方法来消除尾延迟现象。于是就不知道怎么办了，既然不能启动新的线程，那么请求就只能是顺序执行，如果尾延迟的最大值要远远大于启动新线程耗费的时间，那么可以考虑重新启动一个线程去解决。