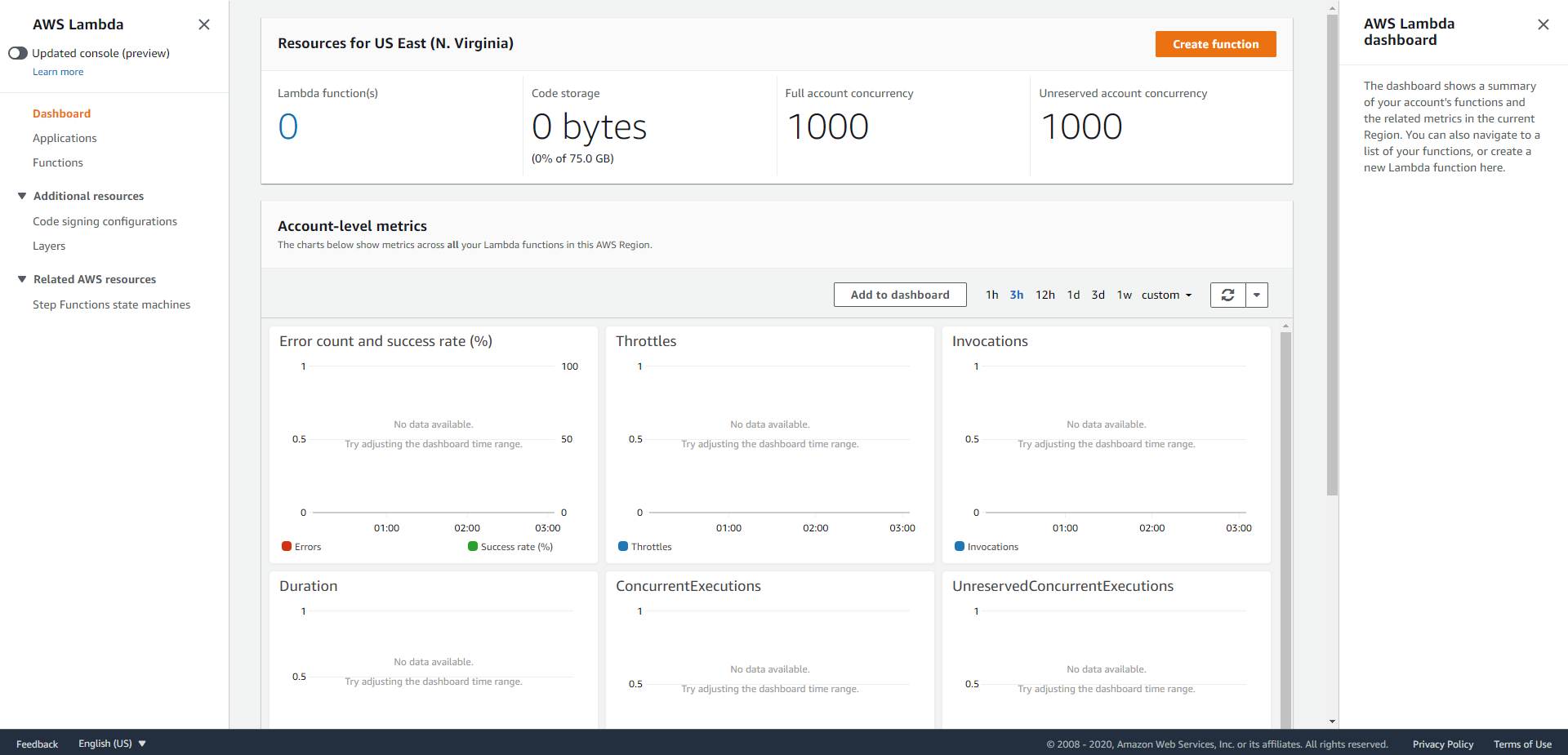
Lambda

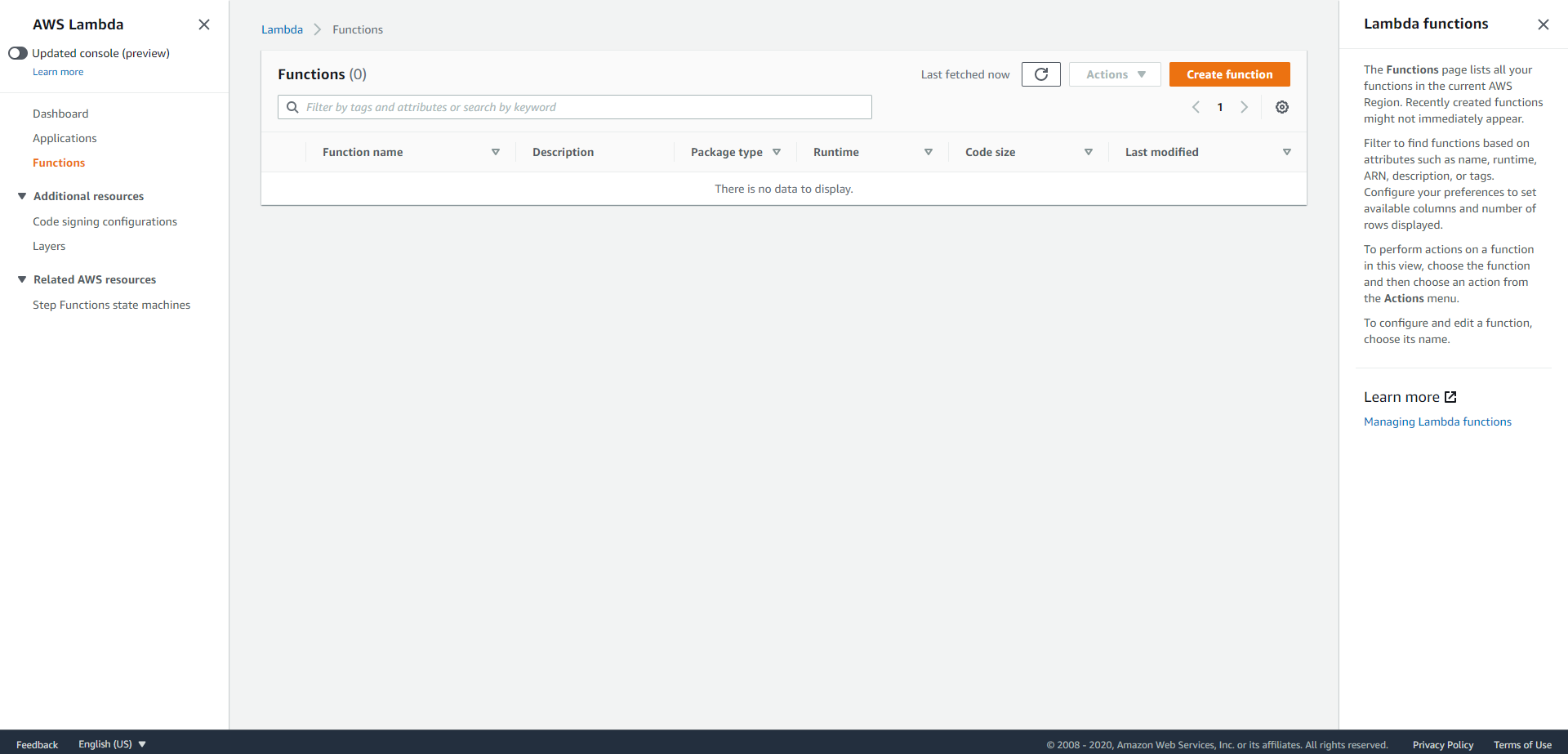
什么是Lambda

AWS Lambda 是一项计算服务，可使您无需预置或管理服务器即可运行代码。AWS Lambda 只在需要时运行您的代码并自动缩放，从每天几个请求到每秒数千个请求。您只需按消耗的计算时间付费 – 代码未运行时不产生费用。借助 AWS Lambda，您几乎可以为任何类型的应用程序或后端服务运行代码，并且不必进行任何管理。AWS Lambda 在可用性高的计算基础设施上运行您的代码，执行计算资源的所有管理工作，其中包括服务器和操作系统维护、容量预置和自动扩展、代码监控和记录。您只需要以 AWS Lambda 支持的一种语言提供您的代码。



函数（Functions）

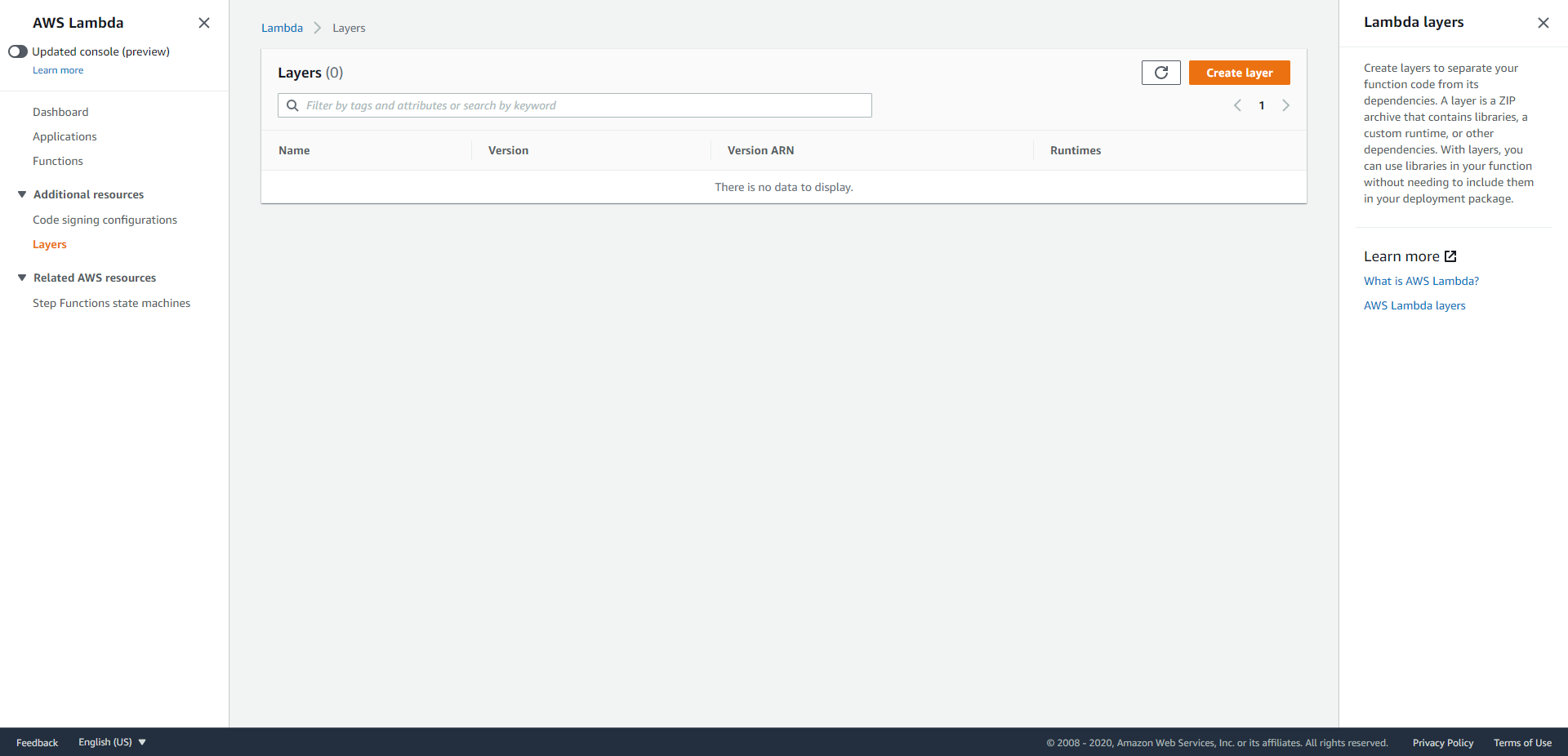
函数 是一个资源，您可以调用它来在 Lambda 中运行您的代码。函数具有处理事件的代码，以及在 Lambda 与函数代码之间传递请求和响应的运行时。您负责提供代码，并且可以使用提供的运行时或创建自己的运行时。



层（Layers）

您可以将 Lambda 函数配置为以层的形式拉入其他代码和内容。层是包含库、自定义运行时或其他依赖项的 ZIP 存档。利用层，您可以在函数中使用库，而不必将库包含在部署程序包中。

使用层可以使您的部署包保持较小，从而使开发变得更轻松。您可以避免在使用函数代码安装和打包依赖项时可能出现的错误。对于 Node.js、Python 和 Ruby 函数，只要将部署程序包保持在 3 MB 以下，就可以在 Lambda 控制台中开发函数代码。



实验：Lambda处理S3存储桶缩略图

1. 创建层，处理缩略图需要Python的Pillow库，因此需要创建层

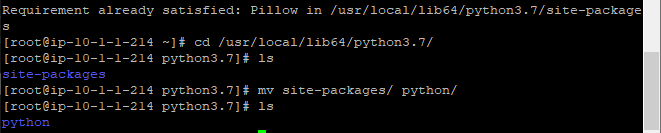
首先启动一台EC2，安装Python3



安装Pillow库



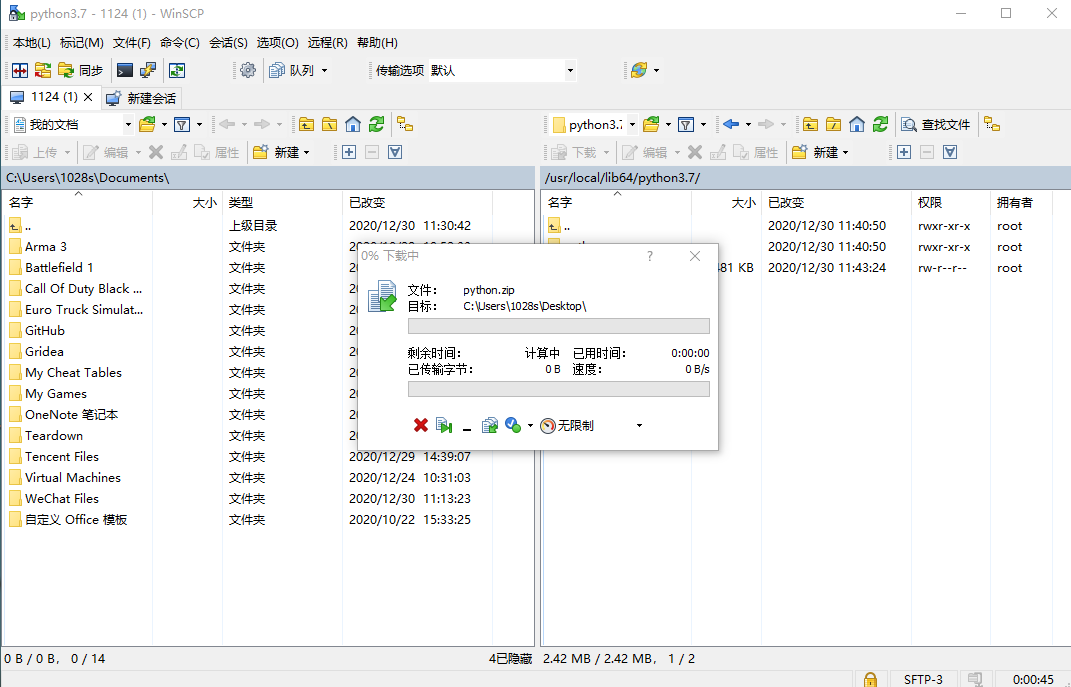
找到Pillow库的位置，将库的上一层目录改名为python



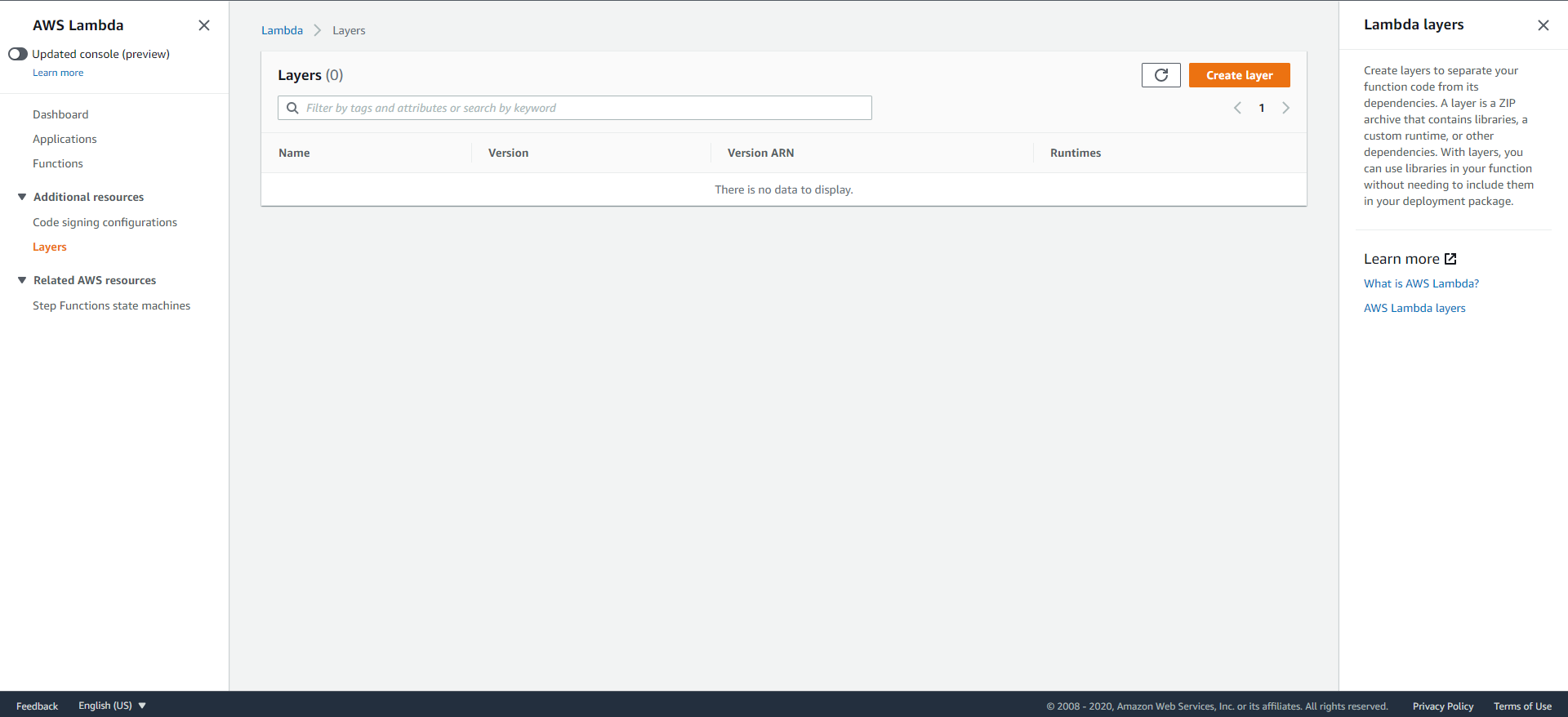
压缩整个目录



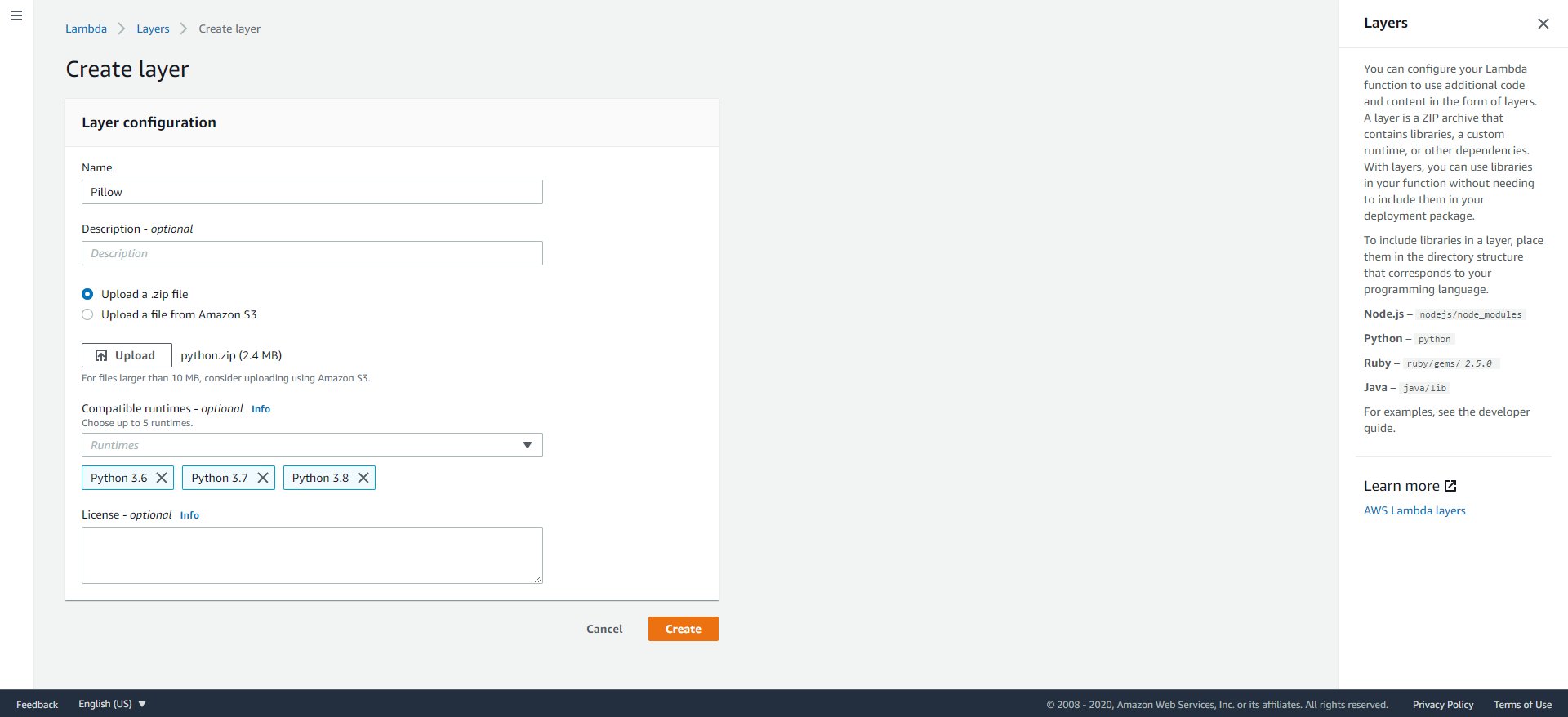
使用WinSCP下载这个zip包到桌面上



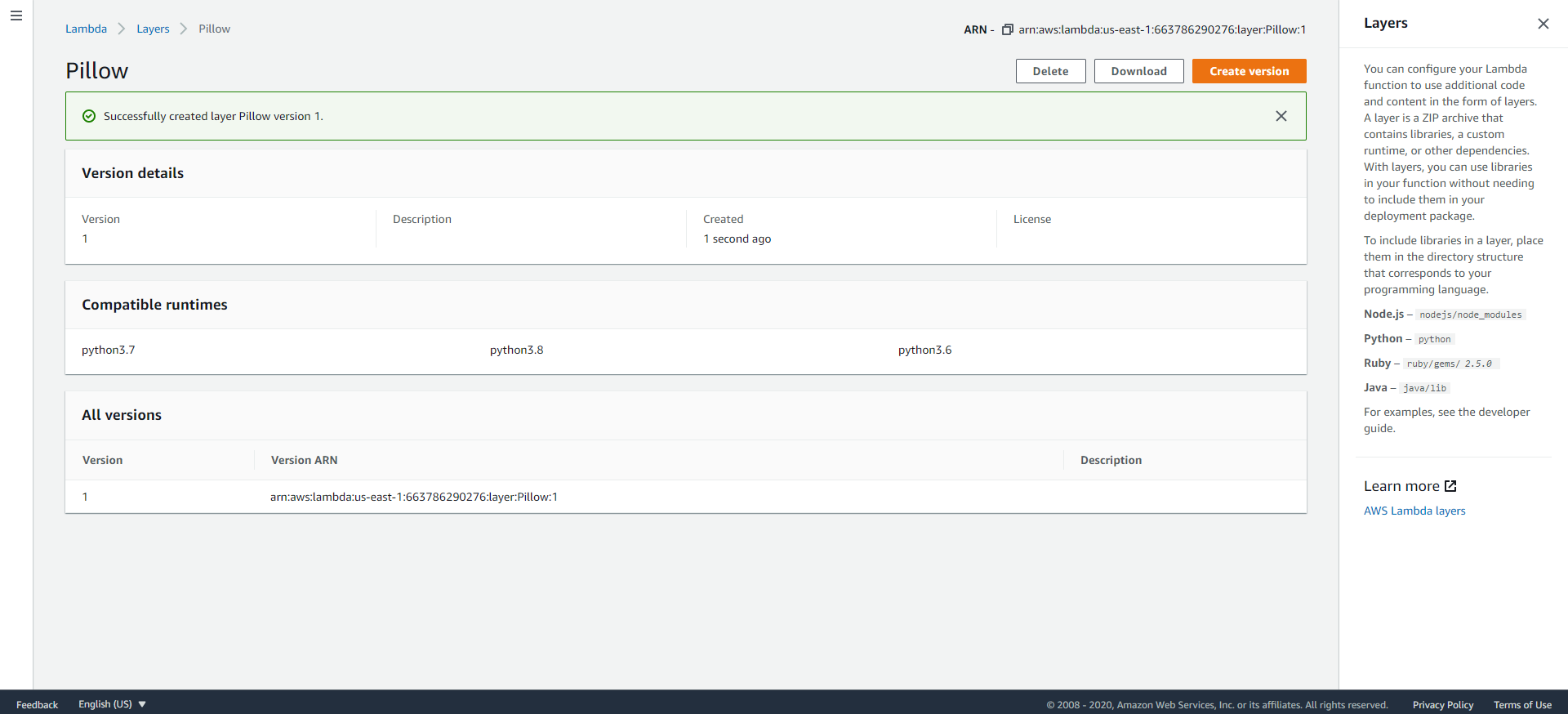
进入Layers界面点击Create Layer



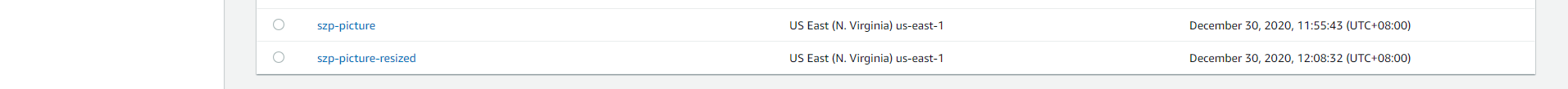
填写名字，上传zip包，并选择兼容的运行环境



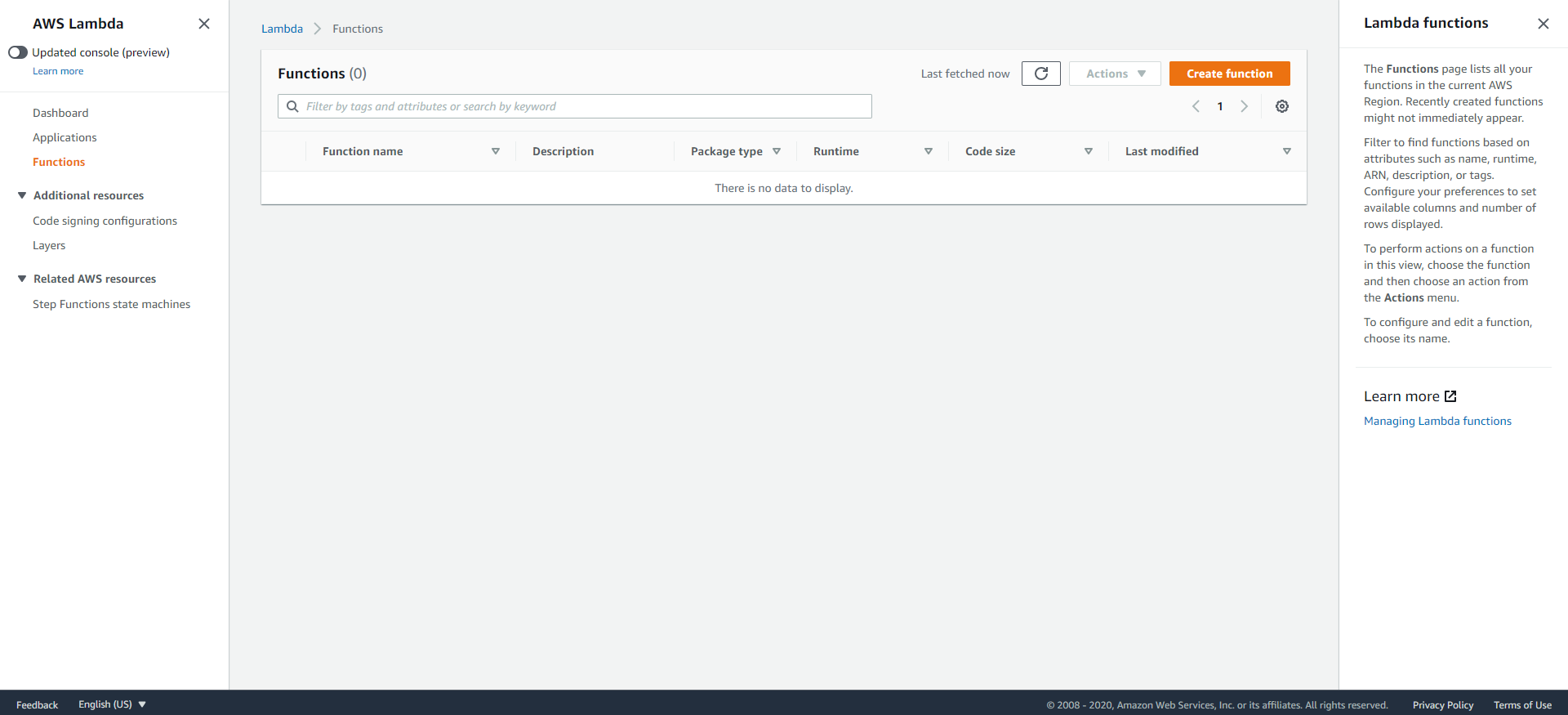
点击创建，层就创建完成了



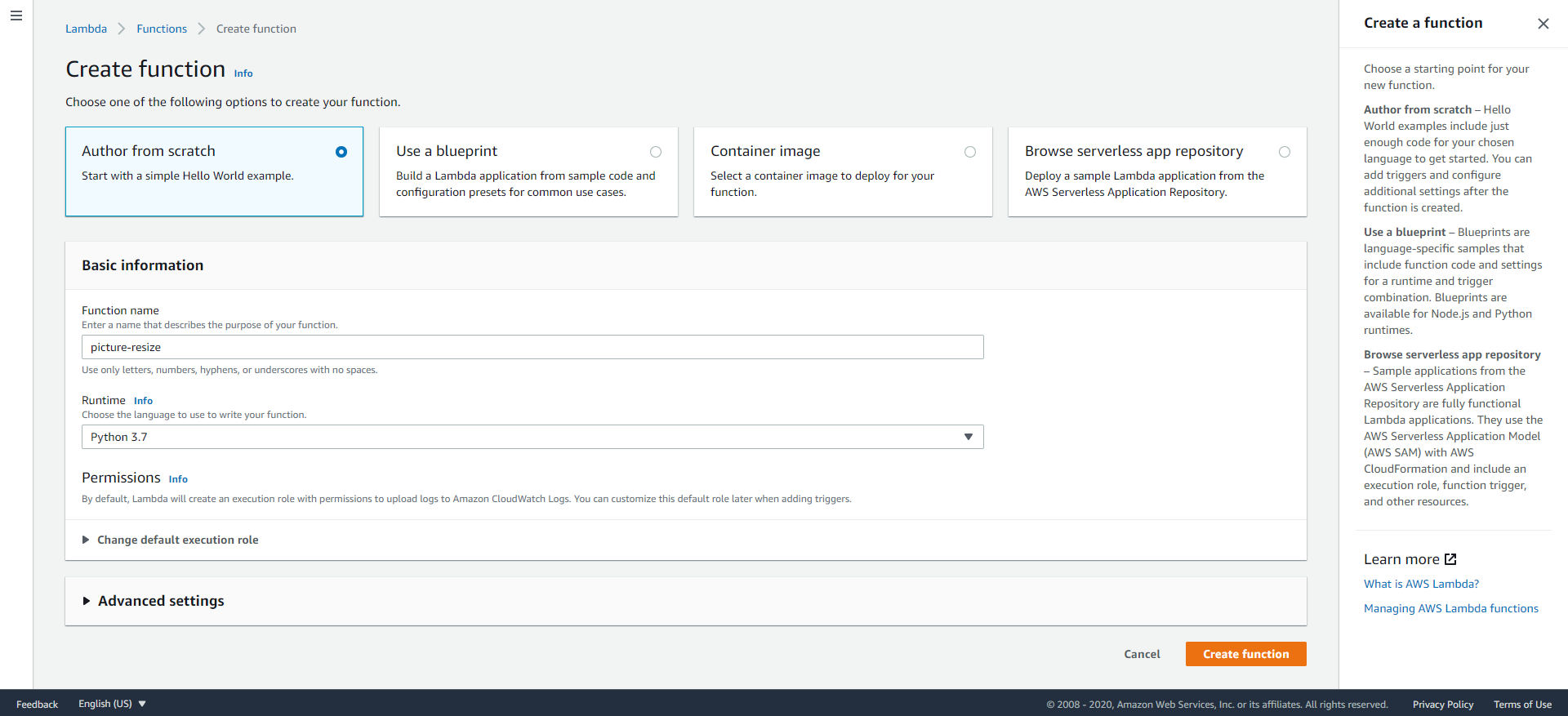
1. 在S3界面创建两个存储桶



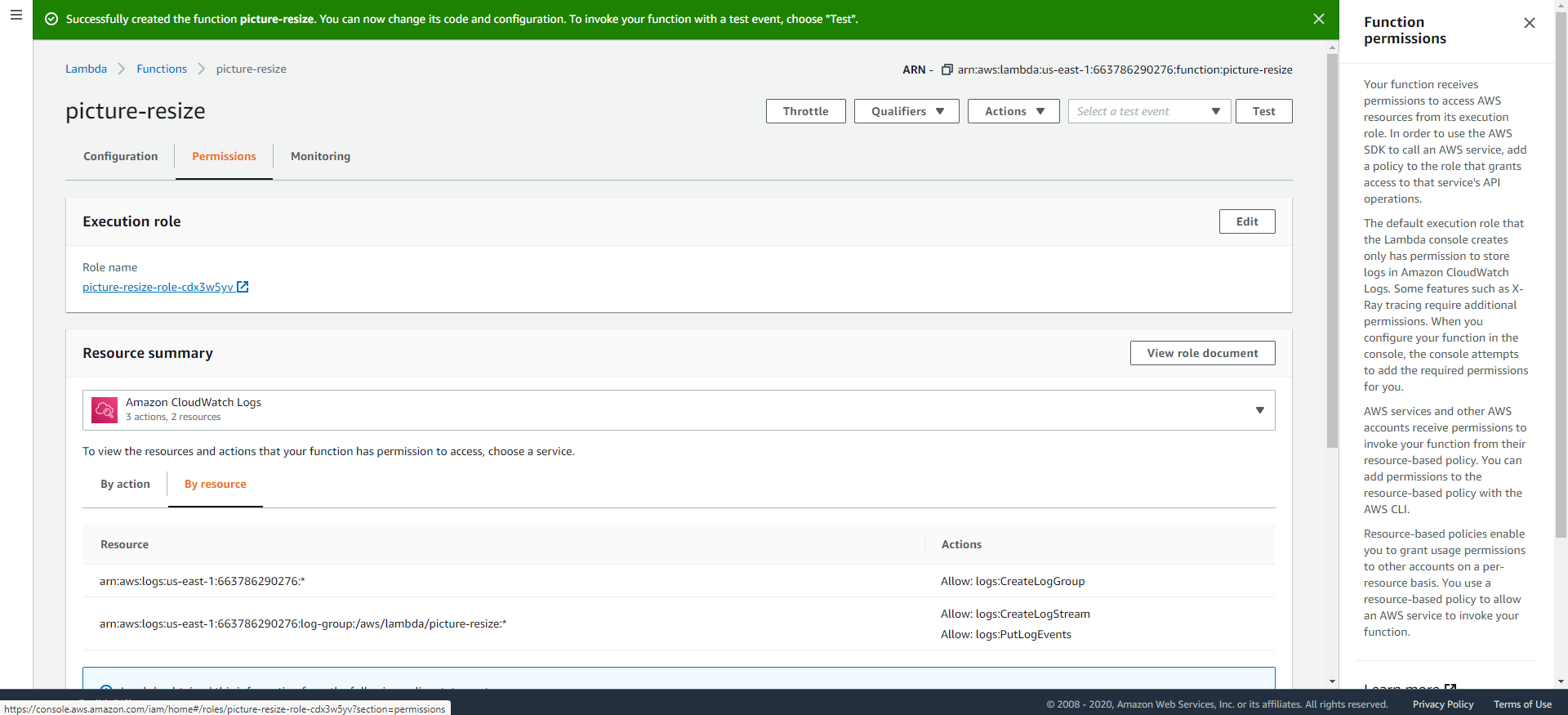
1. 回到Lambda界面，Create function



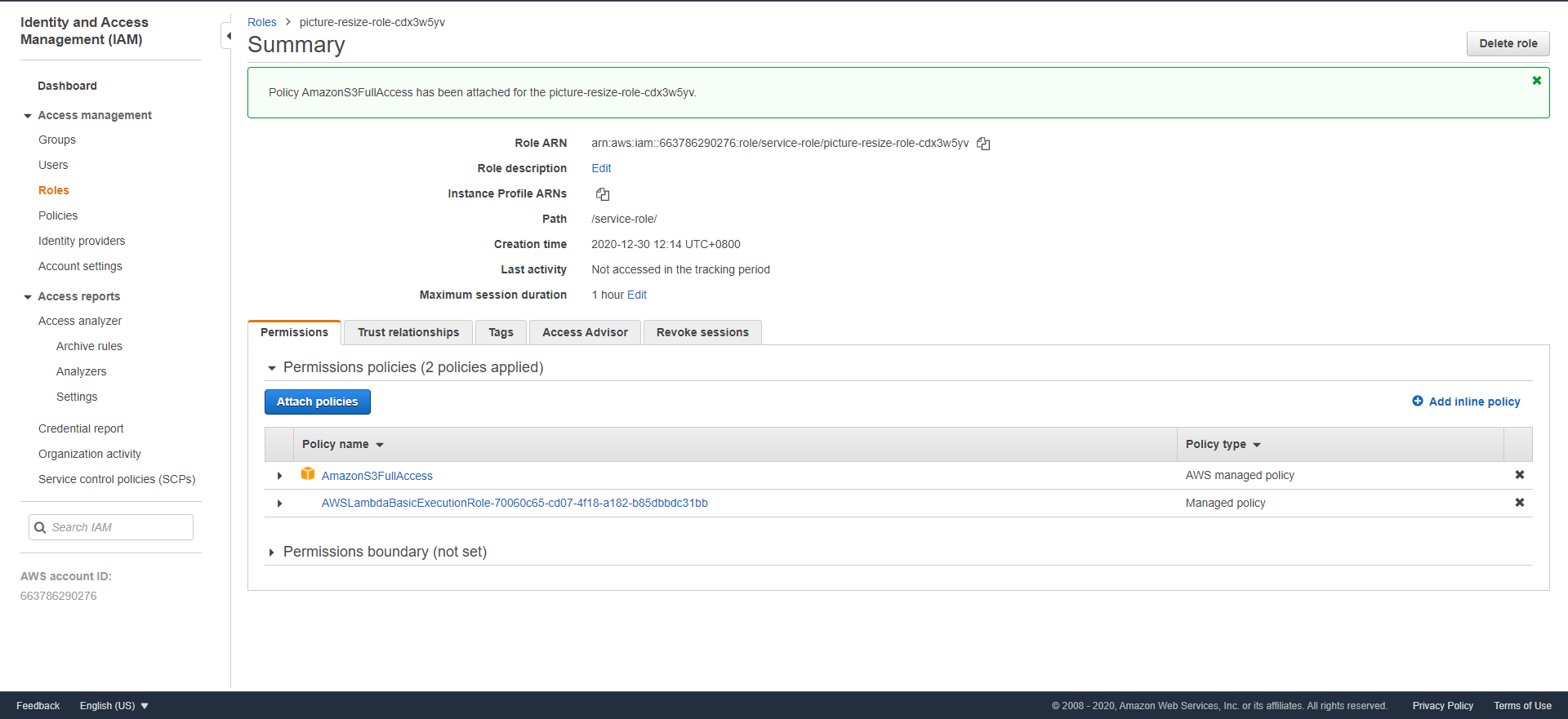
1. 填写名字，选择运行环境



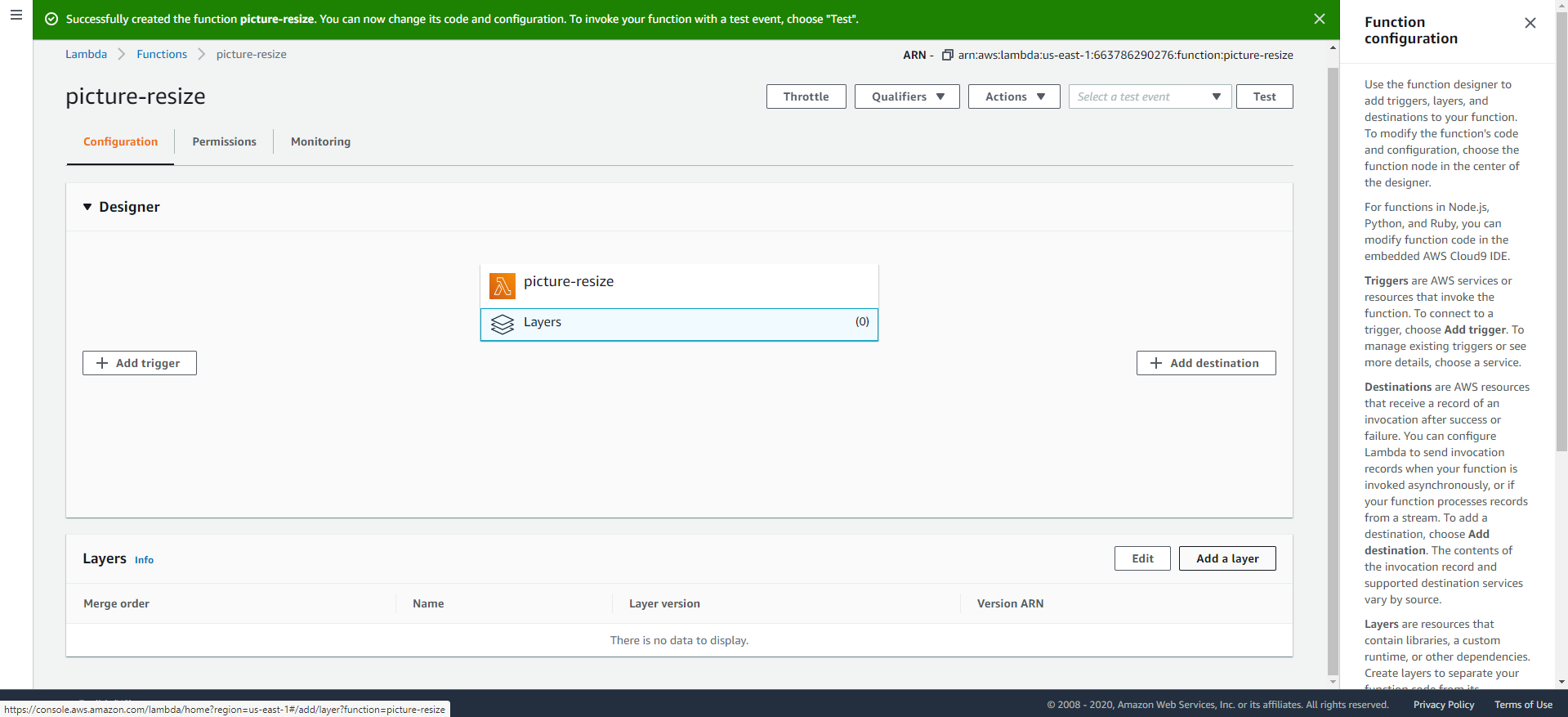
5、点击Permissions，选择Execution role下面的角色



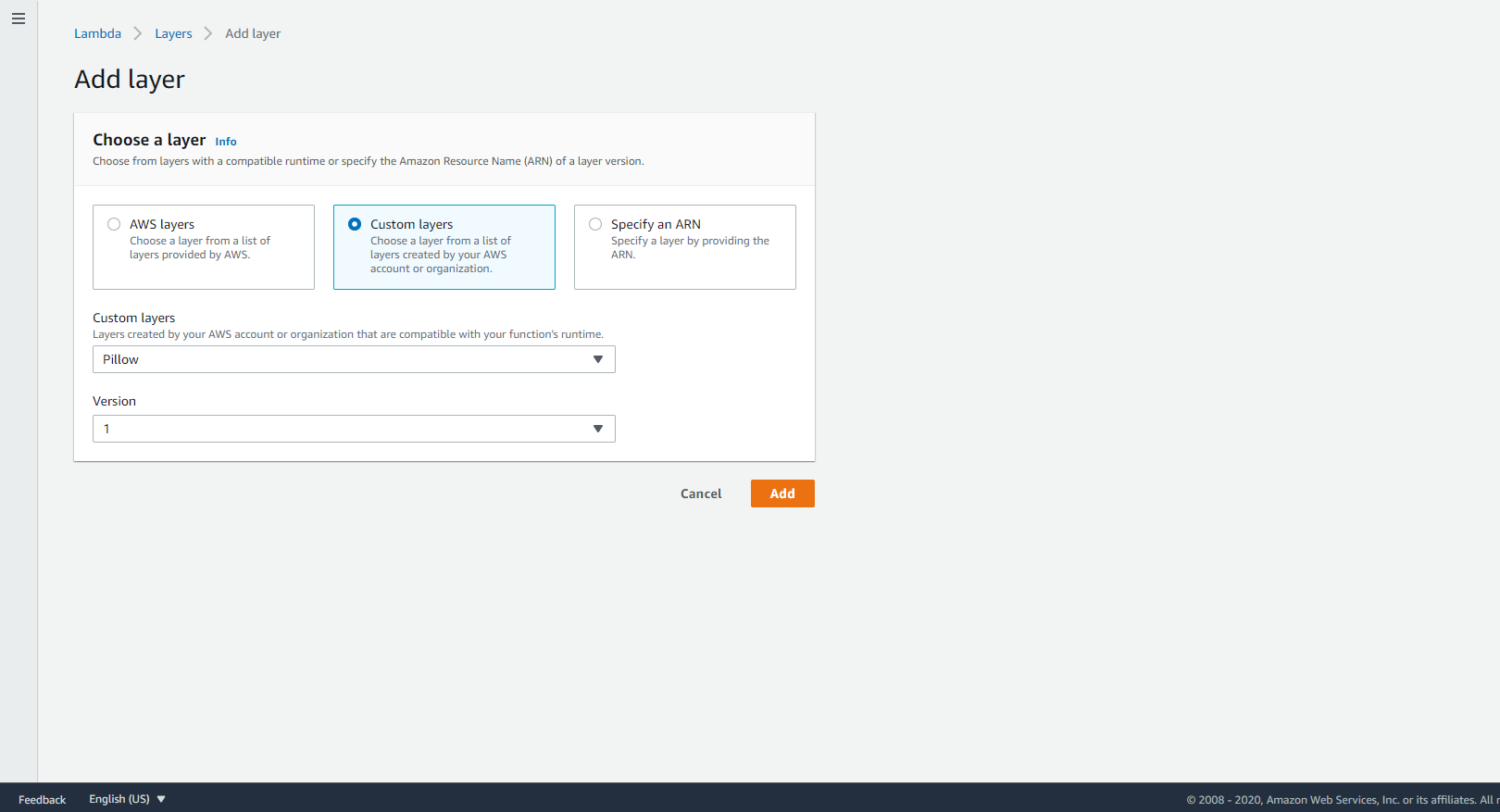
为其赋予S3的执行角色



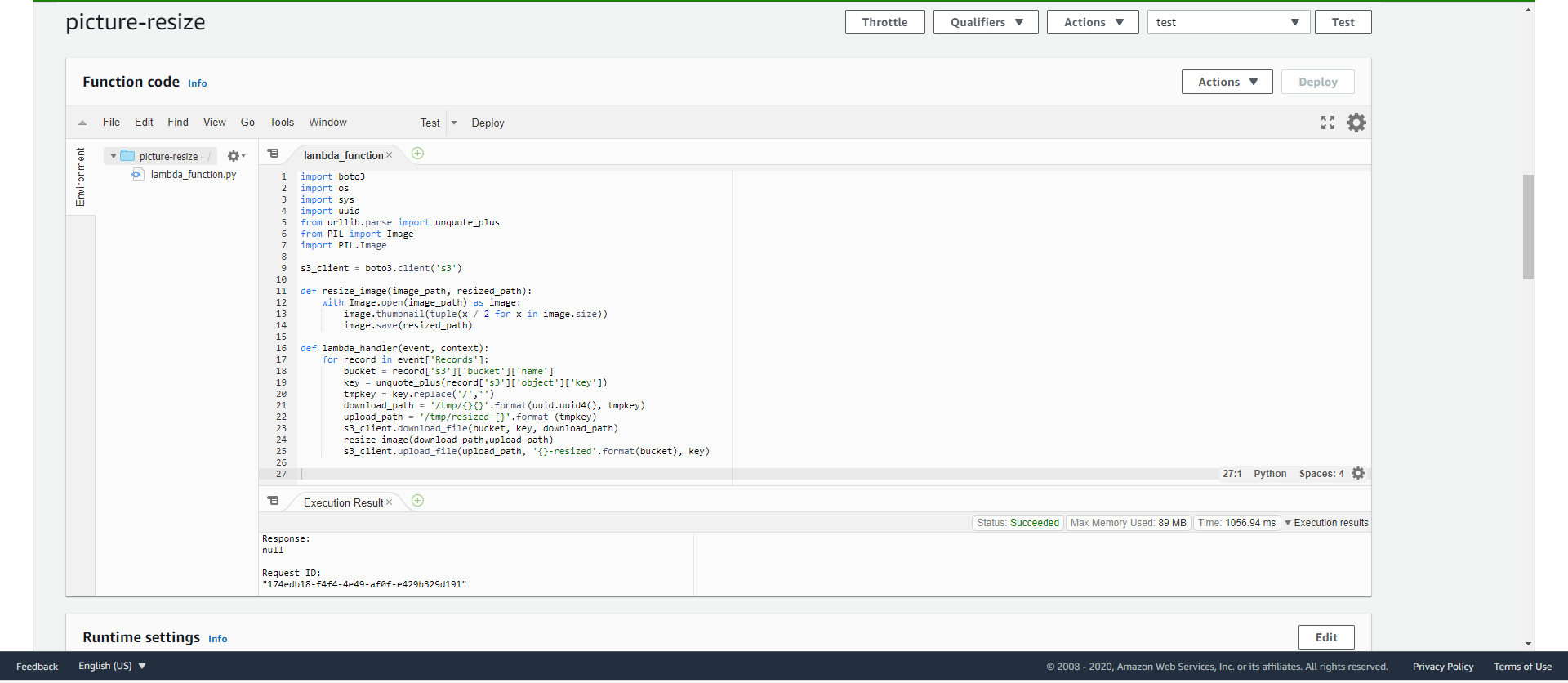
1. 回到Lambda界面，点击Layers->Add a layer



选择Custom layers



1. 开始编写代码



import boto3

import os

import sys

import uuid

from urllib.parse import unquote\_plus

from PIL import Image

import PIL.Image

s3\_client = boto3.client('s3')

def resize\_image(image\_path, resized\_path):

with Image.open(image\_path) as image:

image.thumbnail(tuple(x / 2 for x in image.size))

image.save(resized\_path)

def lambda\_handler(event, context):

for record in event['Records']:

bucket = record['s3']['bucket']['name']

key = unquote\_plus(record['s3']['object']['key'])

tmpkey = key.replace('/','')

download\_path = '/tmp/{}{}'.format(uuid.uuid4(), tmpkey)

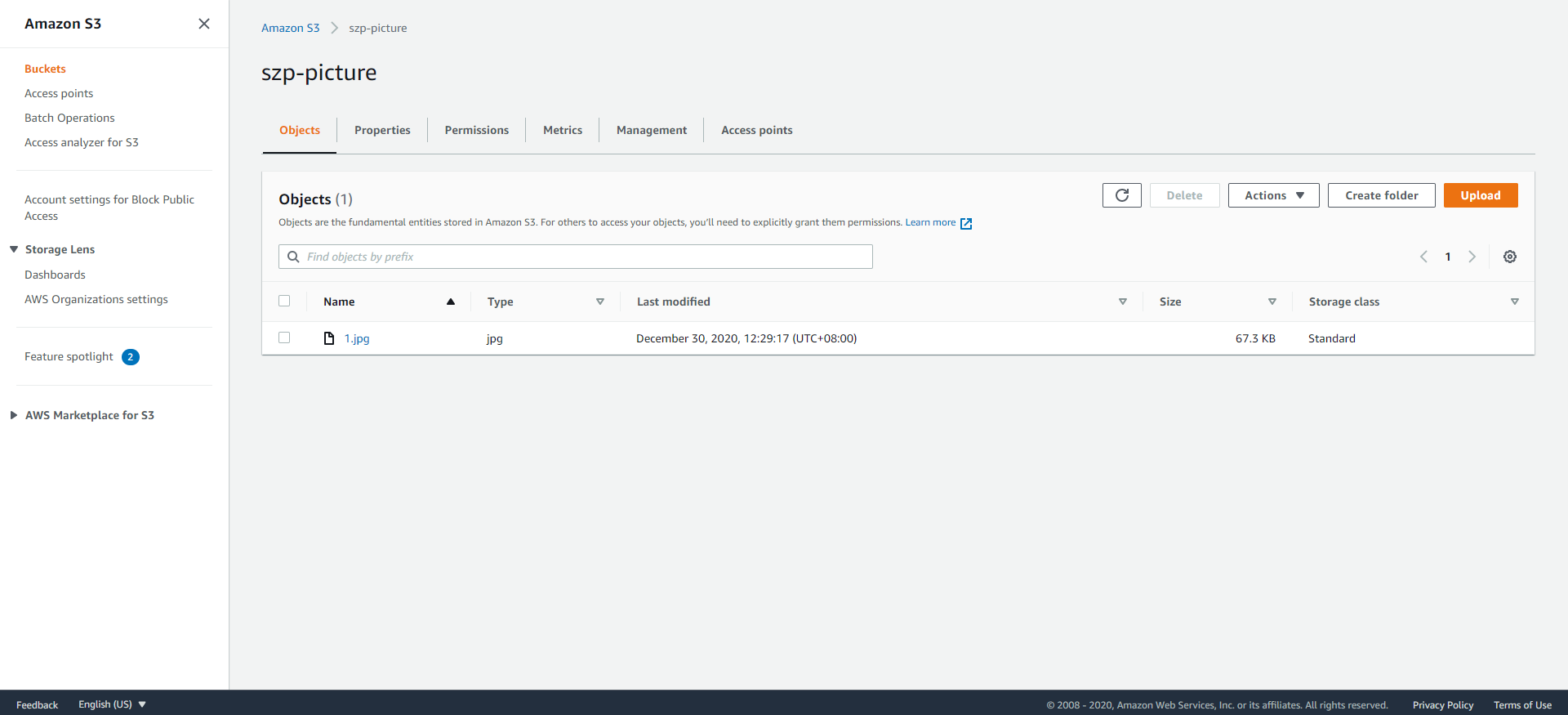
upload\_path = '/tmp/resized-{}'.format (tmpkey)

s3\_client.download\_file(bucket, key, download\_path)

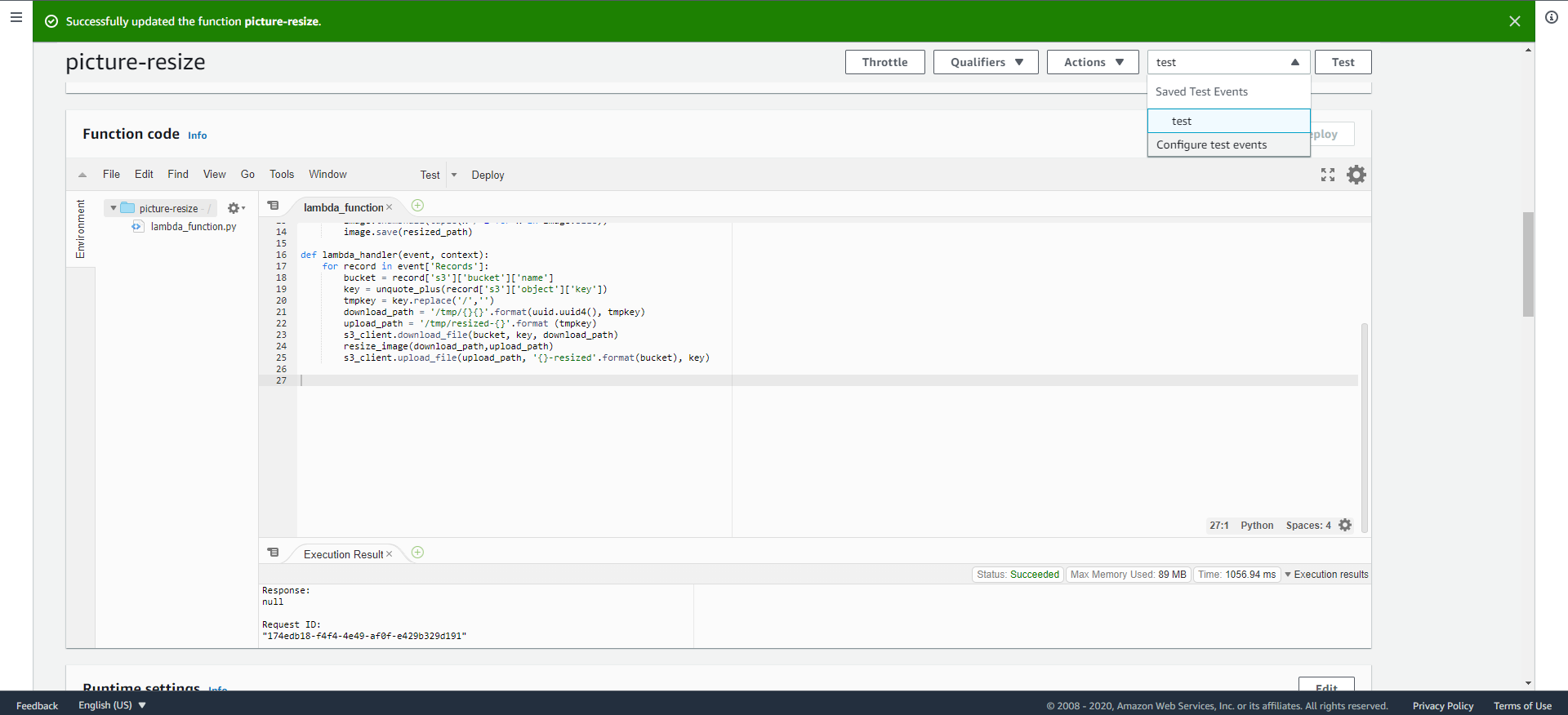
resize\_image(download\_path,upload\_path)

s3\_client.upload\_file(upload\_path, '{}-resized'.format(bucket), key)

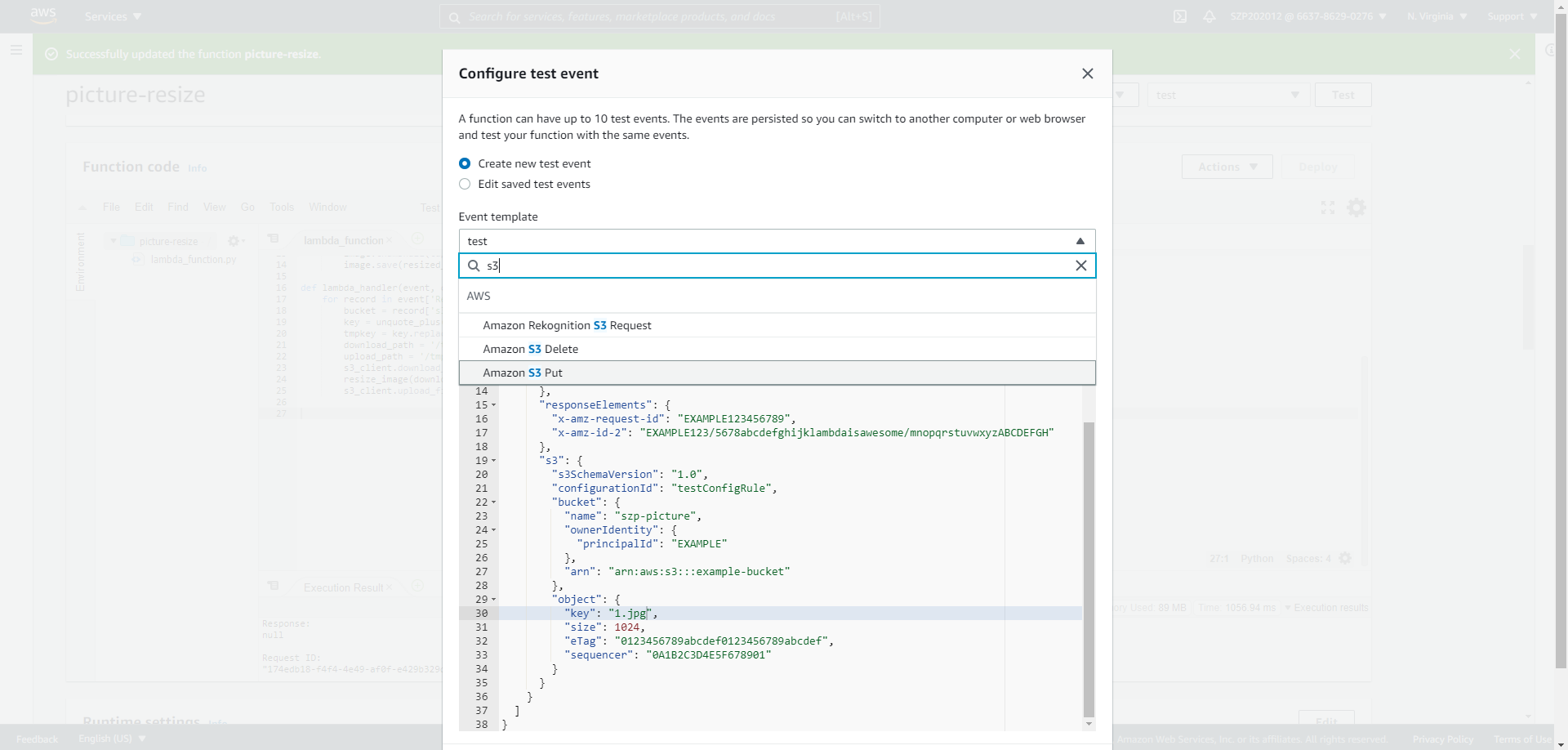
1. 代码编写完成后，找到S3存储桶，上传一张图片



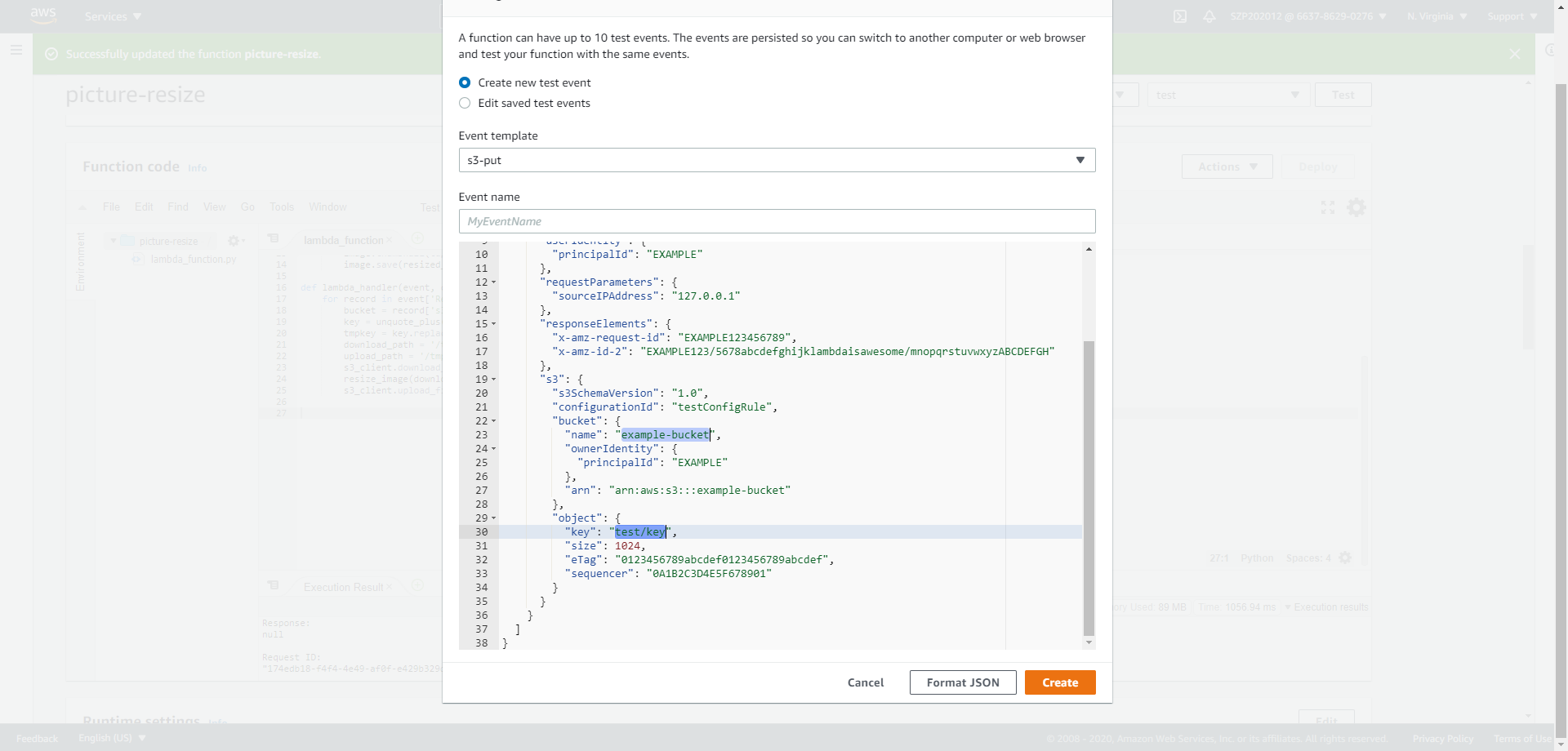
回到lambda代码编写界面点击上方工具栏，选择Configure test events



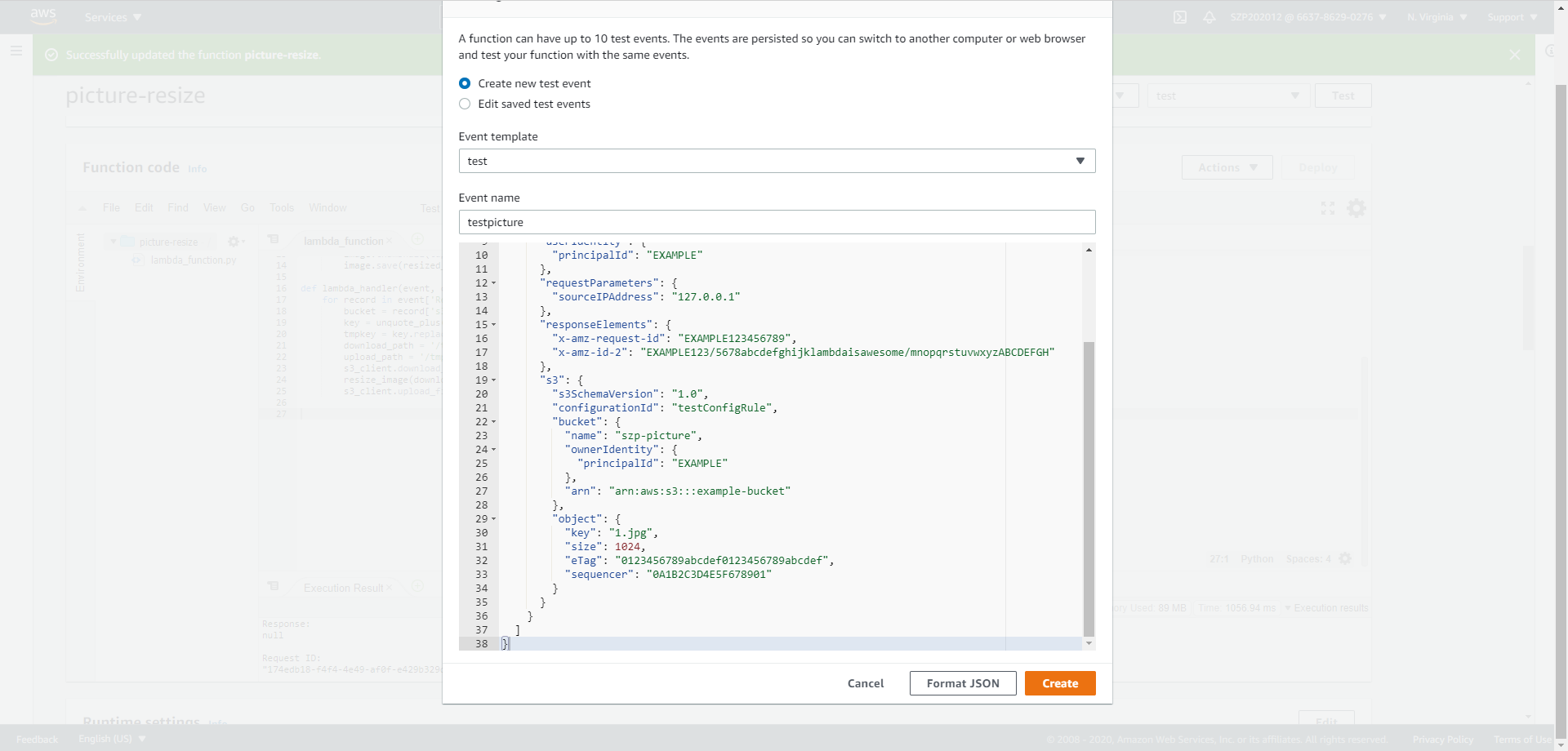
找到s3 put



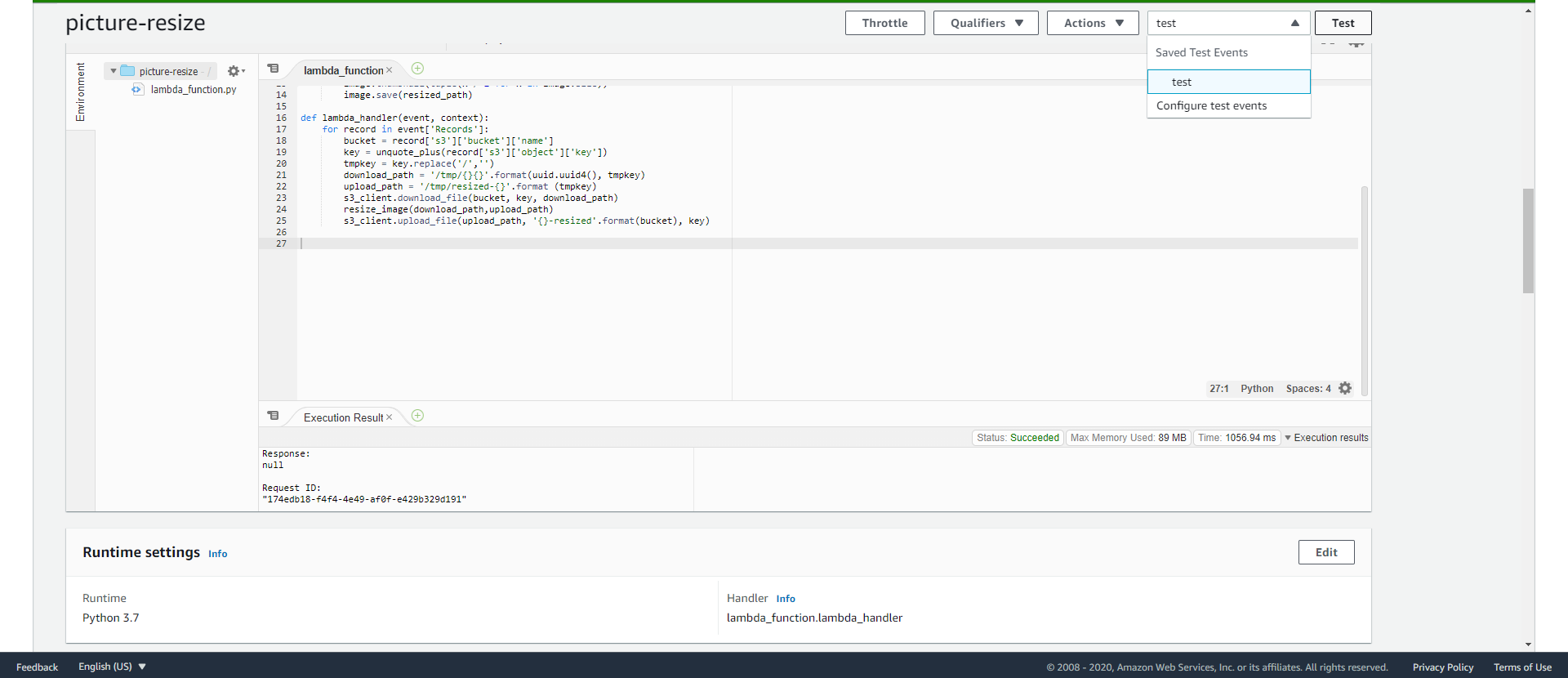
修改name和Key，name是存储桶的名字，key是上传的图片的名字



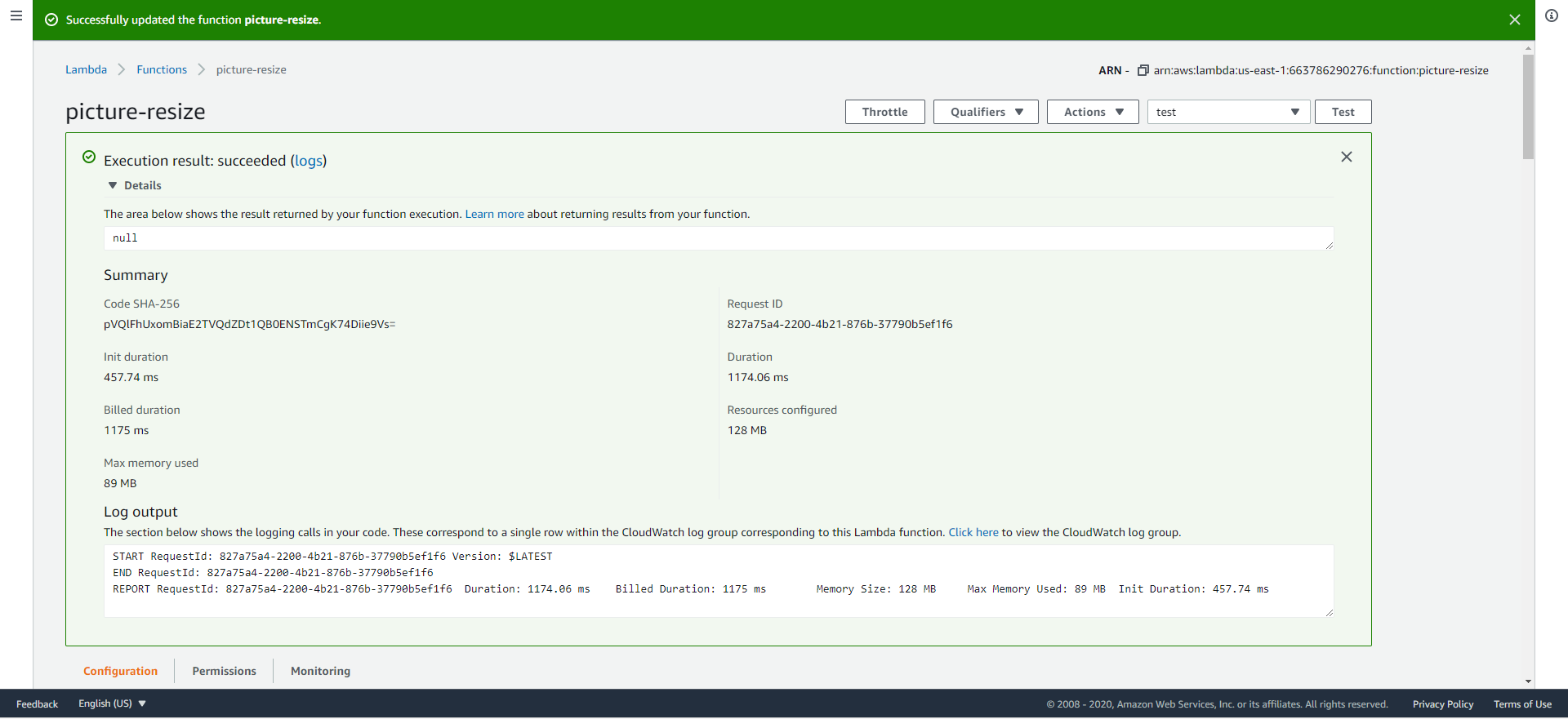
写测试事件的名字，点击Create



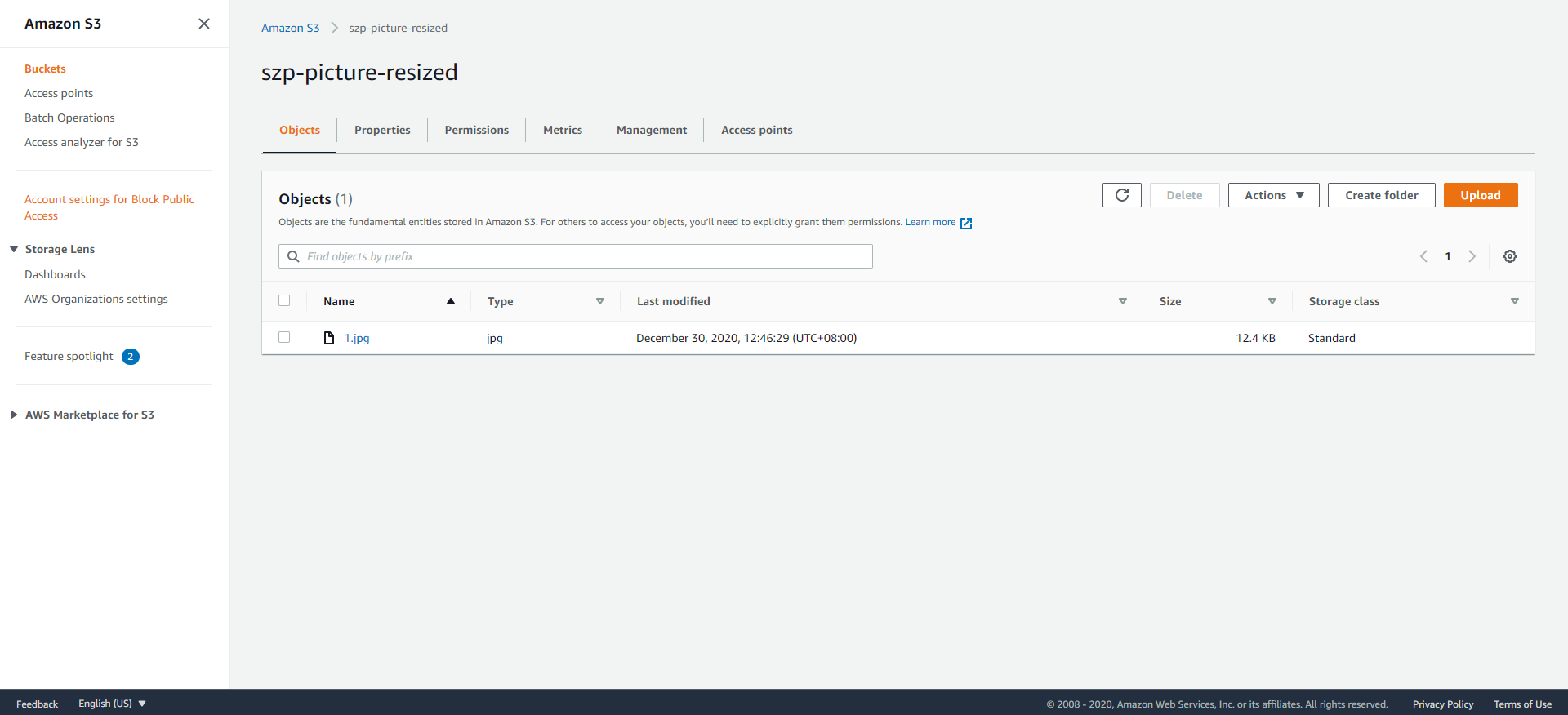
8、在工具栏选择刚创建的测试事件，点击test



发现上方出现一个绿色框，说明测试成功

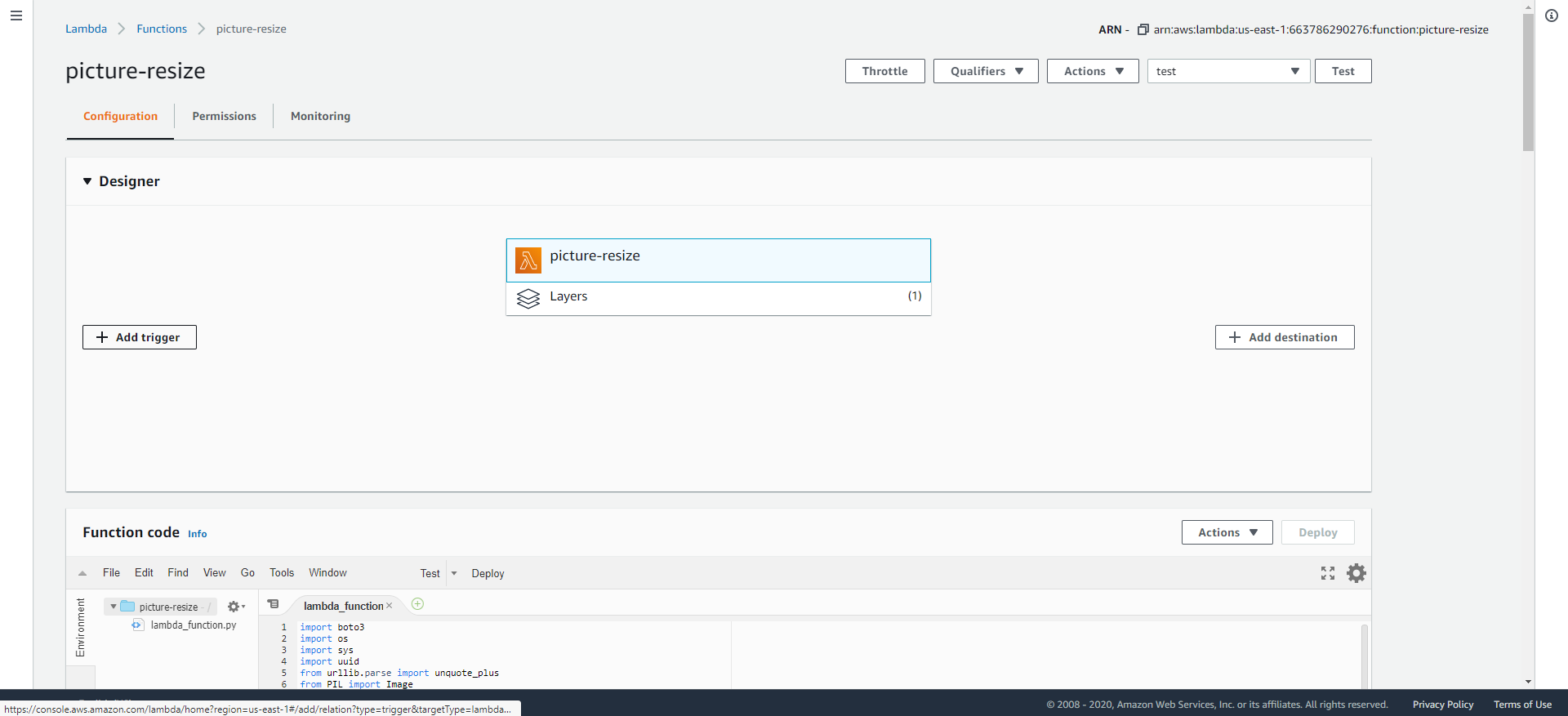


去看另一个“-resize”结尾的存储桶，看看是不是出现了一个容量小一点的同名图片

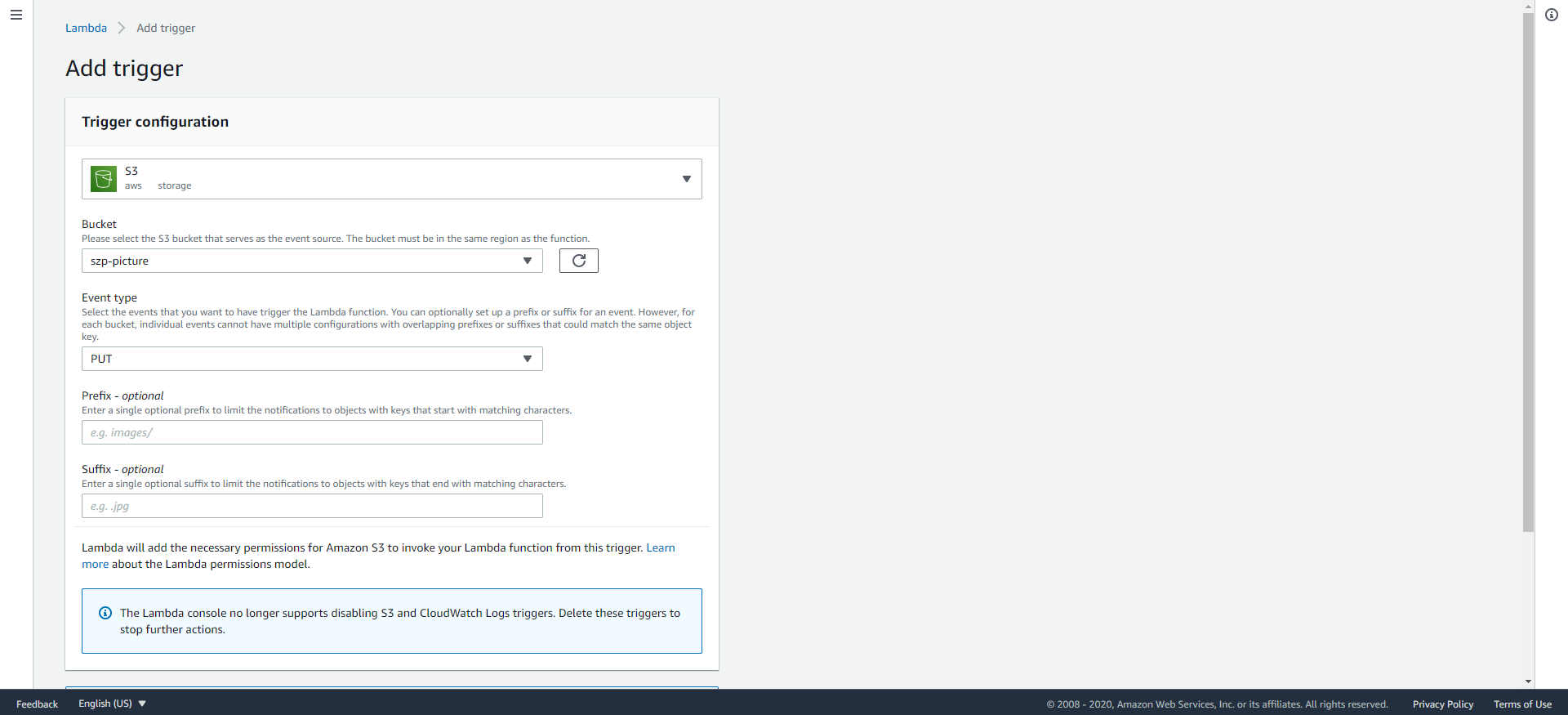


1. 测试成功， 但是现在我们的存储桶状态变化还是不会出发函数运行，我们需要添加触发器

点击Add trigger

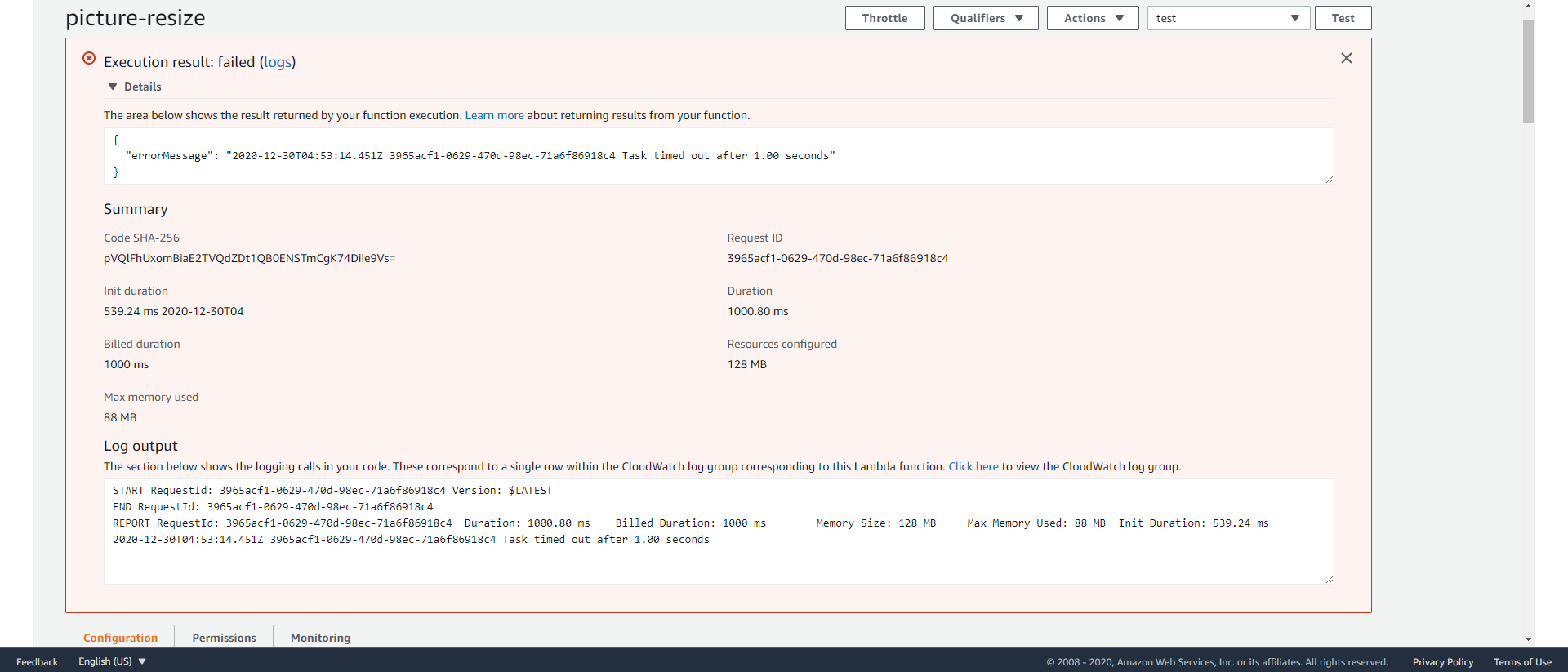


选择s3，Bucket选择自己的存储桶，Event type选择PUT，创建，现在可以试着向存储桶上传文件了



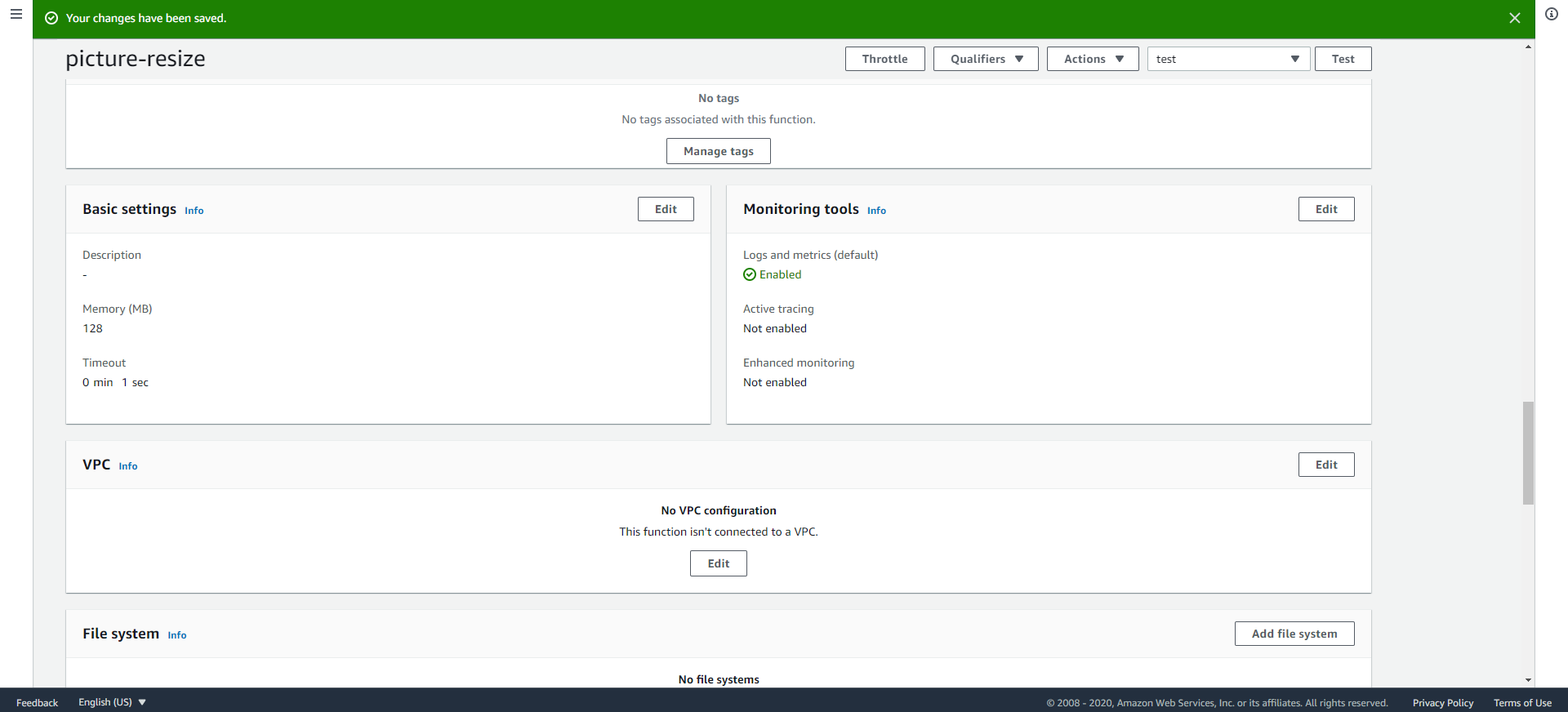
10、实验完成，恭喜！现在往前一个存储同上传图片后，就可以在“-resize”结尾的存储桶中看见被压缩的图片了！

Note：出现了以下错误是什么情况？

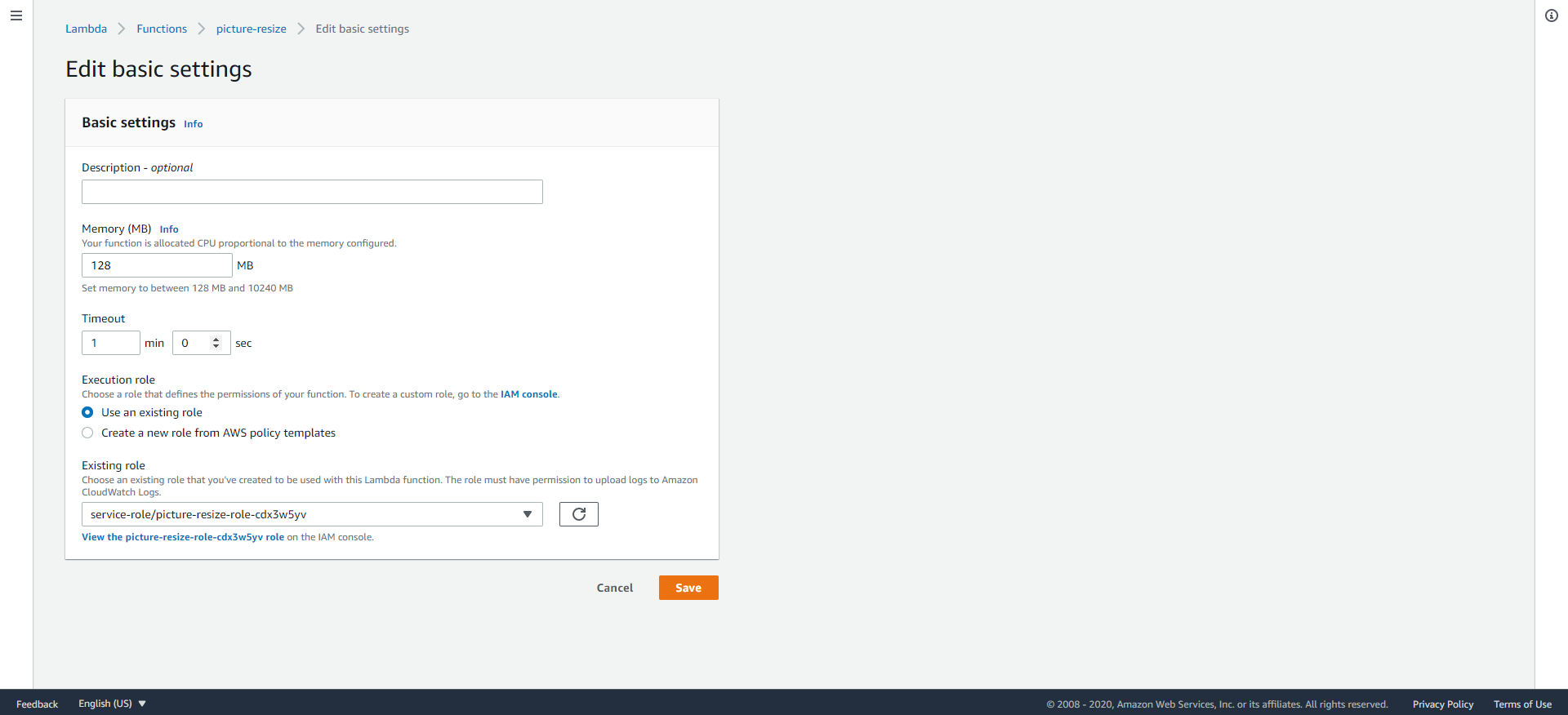


Lambda函数执行是有一个限制时间的，如果在规定时间内函数未执行完成，那么会出现这种情况，在这个实验中会出现这个情况可能是图片太大了等等…当然我梦可以修改这个限制时间

移到代码编写页下方，有个Basic settings，点击edit

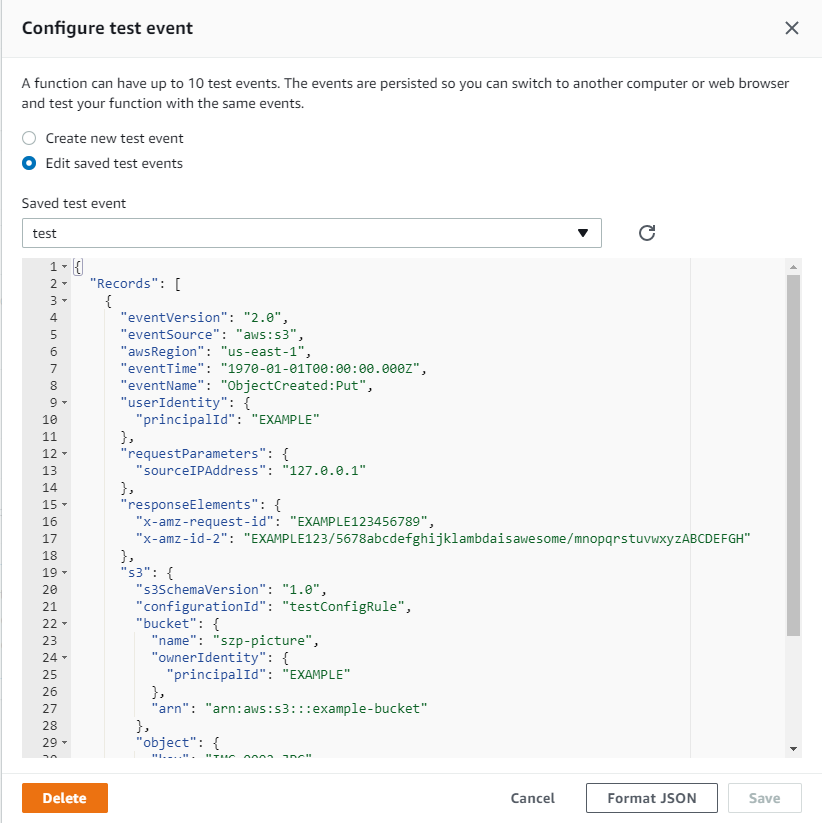


修改这个Time out时间，这个时间默认是3秒，点击保存



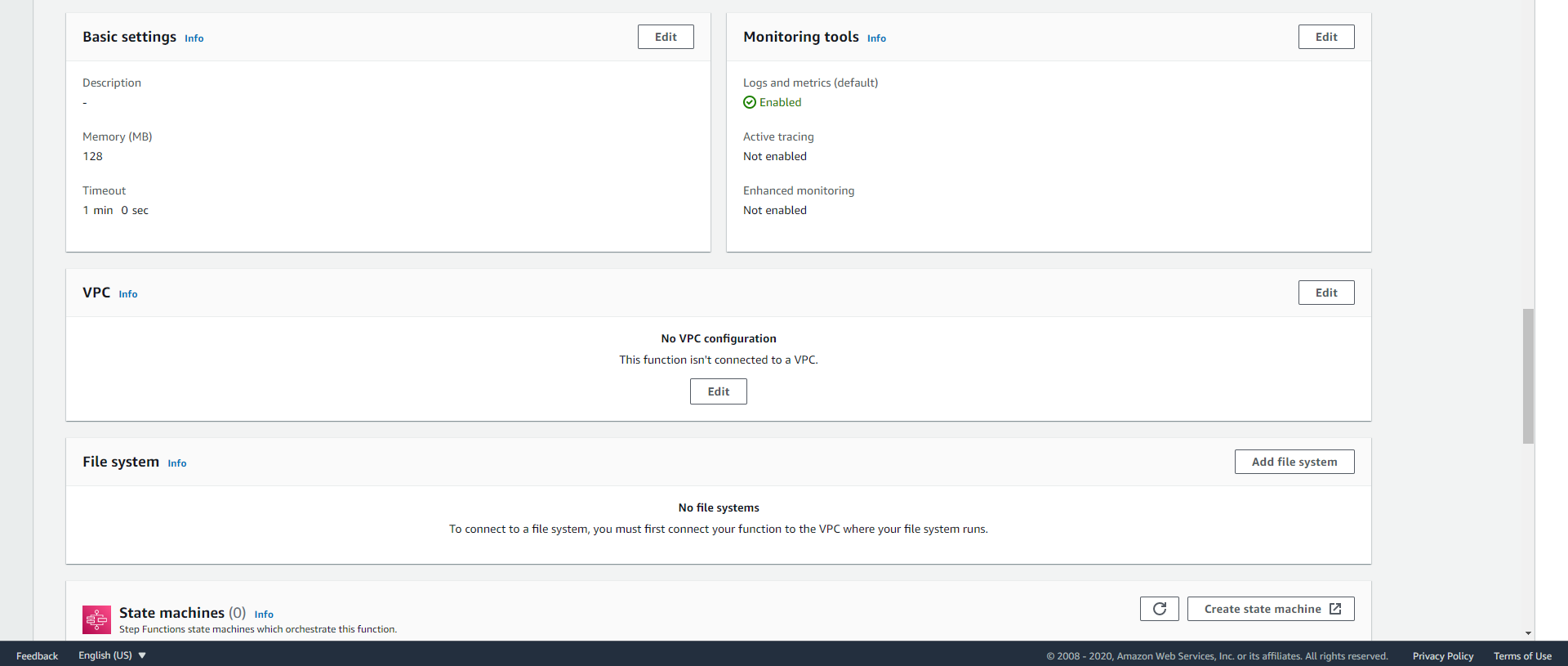
再试一下看看函数能不能执行吧

Note：之前编写的测试事件是什么东西？



AWS的所有组件在使用时都会产生事件，而不同的组件产生的事件是不一样的，有各自的格式，我们实验中编写的测试事件就是S3存储桶在上传文件时后台会产生的事件，我们使用实际不存在的测试事件出来出发函数执行是为了Debug，而后面创建触发器之后，后台真正产生的且符合触发器规则的事件就会触发函数了

Note：为啥Lambda也有VPC设定？



有些时候Lambda所操作的的服务会被放在VPC的私有子网中，比如数据库，这些东西是我们不希望被放在公网中被人随意访问的。但是Lambda作为一个AWS托管组件被放在AWS的网络之中，这使得它平时并不能访问那些在VPC中私有的服务。但是我们可以通过将Lambda也部署到私网的方式来让它也可以访问那些私网中的服务。