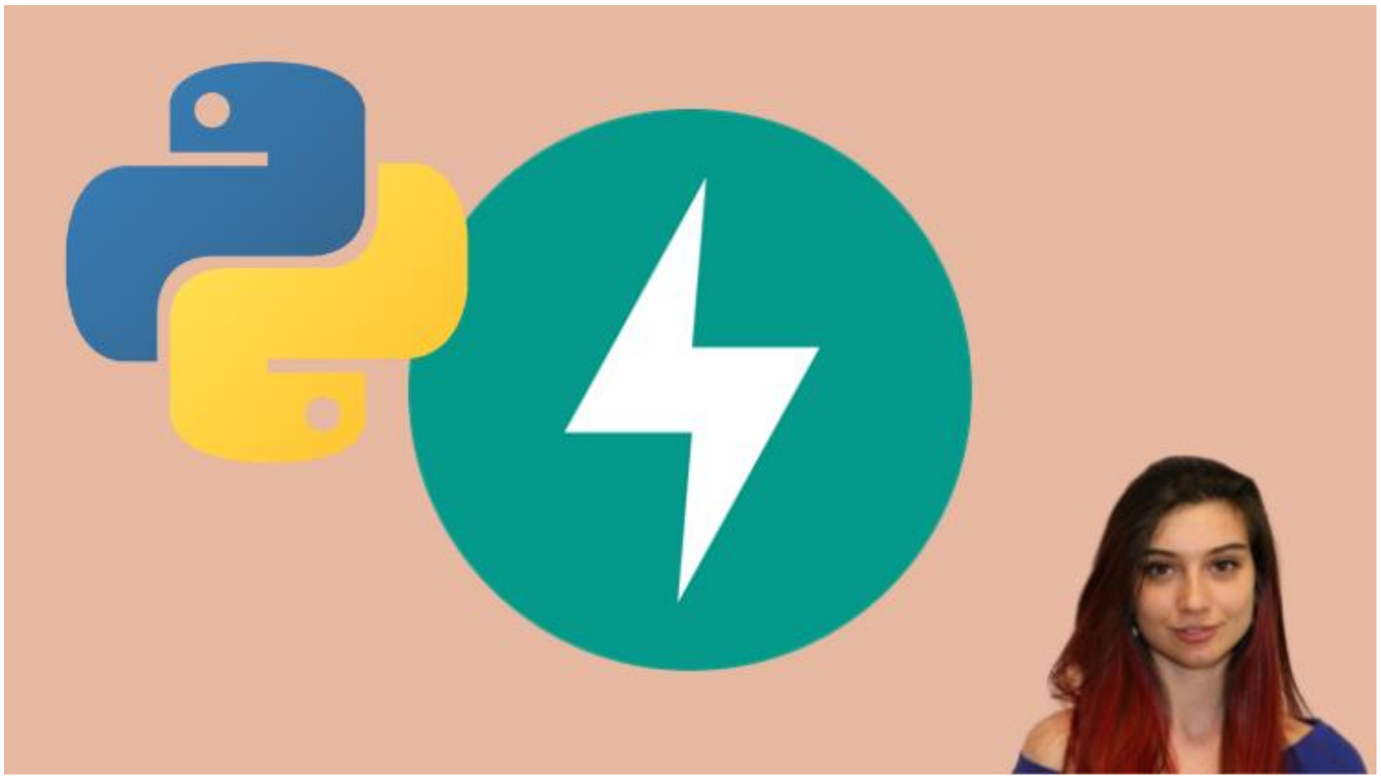
FastAPI REST – Part 6



CODE WITH FINESSE

Ready to go?

Complaint system (course application – Part 2 – AWS S3 bucket)

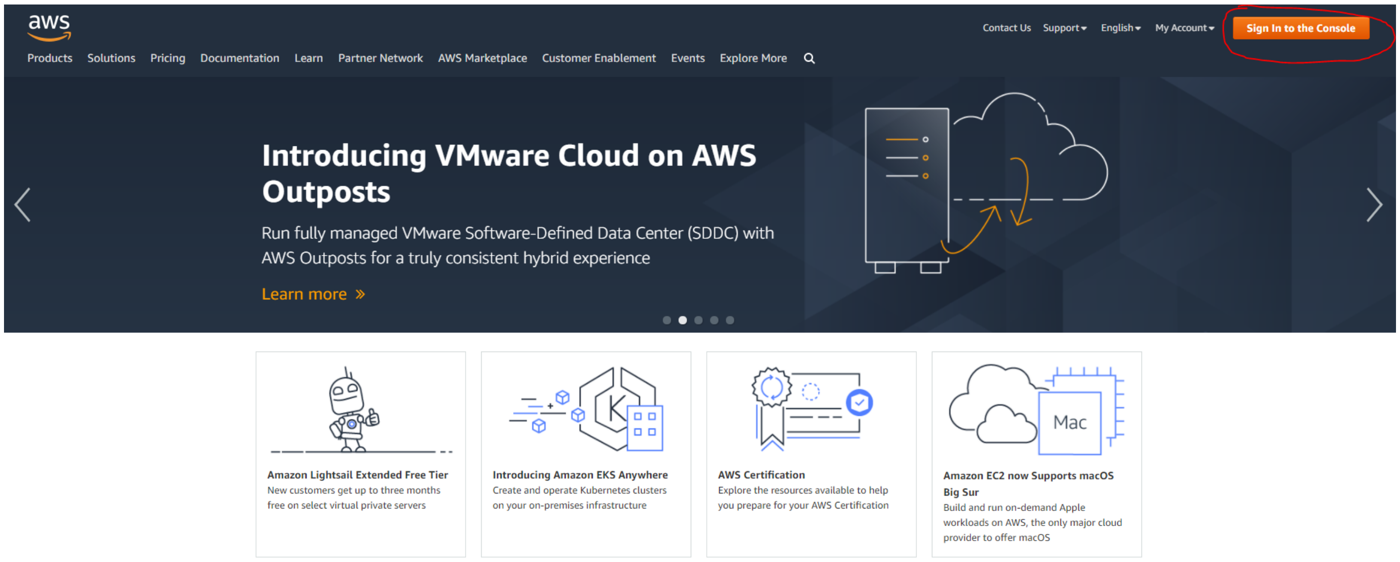
TABLE OF CONTENTS

1. Introduction

Our goal would be to integrate AWS(s3) in our application. For this purpose, we will use boto3 library.
First, we will start with S3 by installing boto3 and connecting our account. Then we will upload the photos from complainers for their expenses.
In the next part, we will integrate Wise to reimburse our complainers for their claim if the approver approves their complaint.
We will continue to develop our architecture and rely on clean code and structured files.

2. Set up AWS account

First, we need to set up an AWS account. You can skip this step if you already have one.
Go to https://aws.amazon.com/ and click the **Sign in to the Console**



Introducing VMware Cloud on AWS Outposts

Run fully managed VMware Software-Defined Data Center (SDDC) with AWS Outposts for a truly consistent hybrid experience

Learn more »

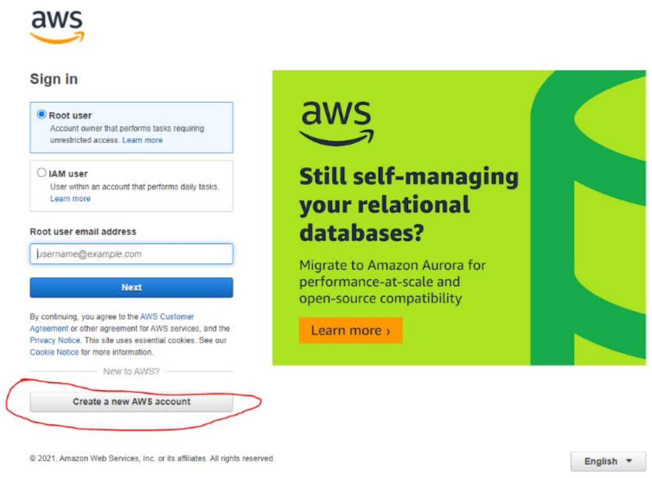
Amazon Lightsail Extended Free Tier
New customers get up to three months free on select virtual private servers

Introducing Amazon EKS Anywhere
Create and operate Kubernetes clusters on your on-premises infrastructure

AWS Certification
Explore the resources available to help you prepare for your AWS Certification

Amazon EC2 now Supports macOS Big Sur
Build and run on-demand Apple workloads on AWS, the only major cloud provider to offer macOS

Then **Create a new AWS account**



Sign in

[✓ Root user]
Account owner that performs tasks requiring unrestricted access. Learn more

IAM user
User within an account that performs daily tasks. Learn more

Root user email address

username@example.com

By continuing, you agree to the AWS Customer Agreement or other agreement for AWS services, and the Privacy Notice. This site uses essential cookies. See our Cookie Notice for more information.

New to AWS?

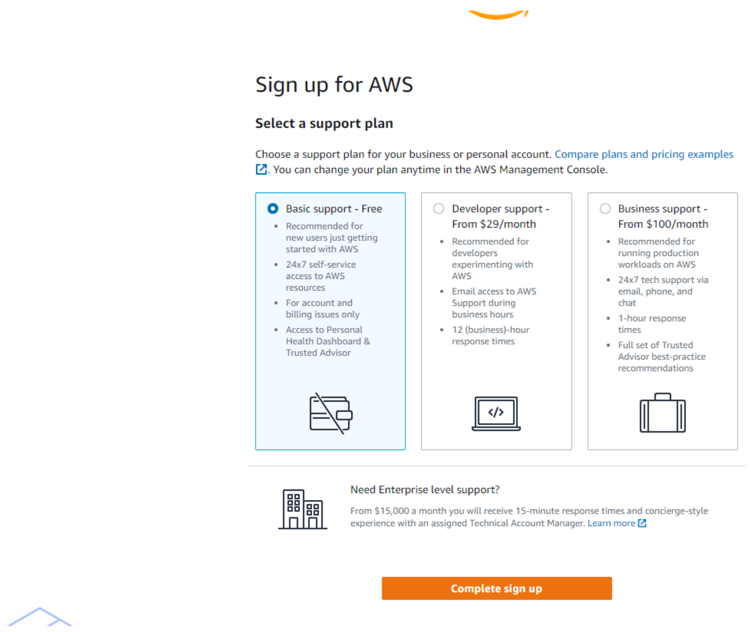
Create a new AWS account

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Still self-managing your relational databases?

Migrate to Amazon Aurora for performance-at-scale and open-source compatibility

For security reasons, we cannot show the whole registration process because sensitive information would be exposed. You have to follow the 5 steps registration process. AWS will charge you 1 USD dollar for the registration. Next, you have to verify your phone number by entering the 4-digit code they will send you as the last step. When you finish the registration process successfully you will see something similar to:



Sign up for AWS

Select a support plan

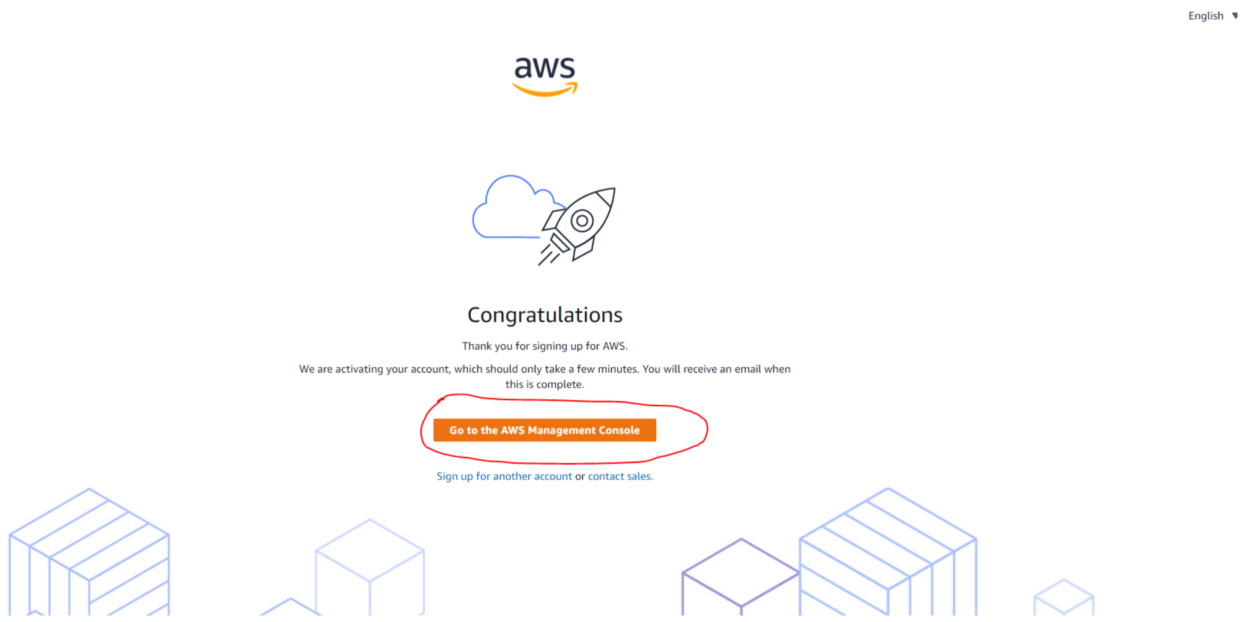
Choose a support plan for your business or personal account. Compare plans and pricing examples. You can change your plan anytime in the AWS Management Console.

- [✓] Basic support - Free
  - Recommended for new users just getting started with AWS
  - 24x7 self-service access to AWS resources
  - For account and billing issues only
  - Access to Personal Health Dashboard & Trusted Advisor
- Developer support - From $29/month
  - Recommended for developers experimenting with AWS
  - Email access to AWS Support during business hours
  - 12 (business)-hour response times
- Business support - From $100/month
  - Recommended for running production workloads on AWS
  - 24x7 tech support via email, phone, and chat
  - 1-hour response times
  - Full set of Trusted Advisor best-practice recommendations

Need Enterprise level support?

From $15,000 a month you will receive 15-minute response times and concierge-style experience with an assigned Technical Account Manager. Learn more

Choose the basic plan, so that you do not get any additional charges. Click **Complete sign up**



Congratulations

Thank you for signing up for AWS.

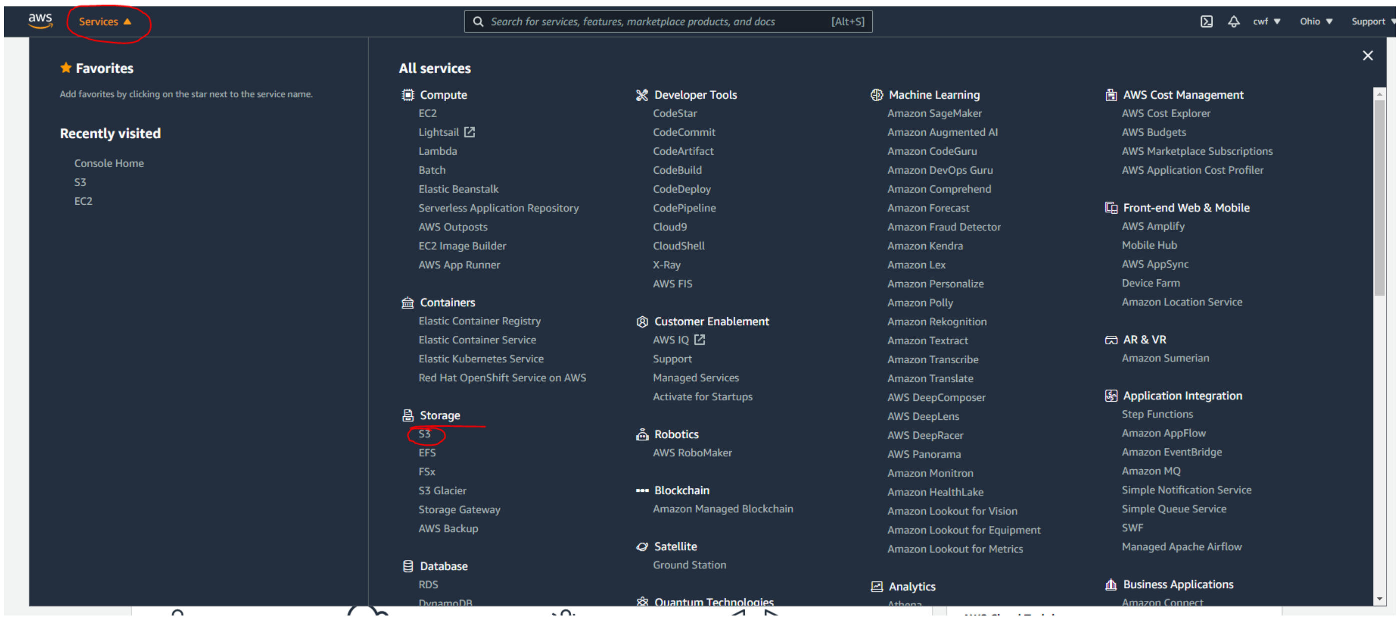
We are activating your account, which should only take a few minutes. You will receive an email when this is complete.

Go to the AWS Management Console
Sign up for another account or contact sales.

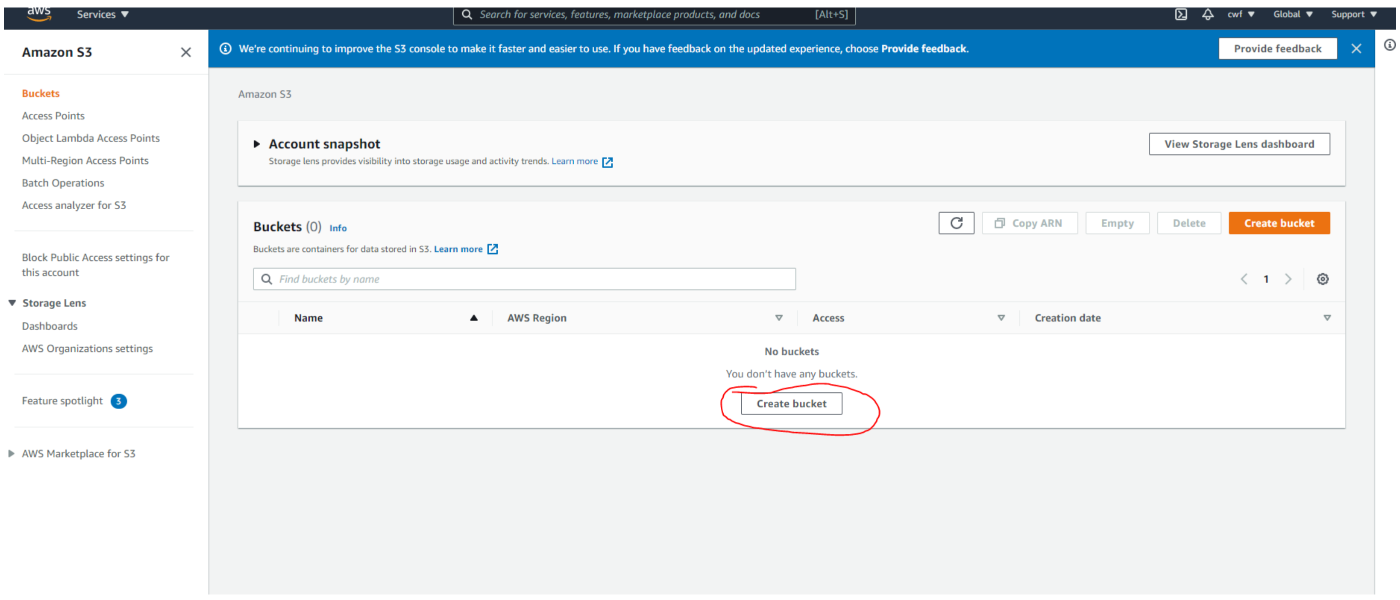
And then Go to the **AWS Management Console**.
Now we have our account. Next step – to set up the bucket!

3. Set up S3

Once you have your AWS account ready and you are in the Console, you have to select **Services** and then **S3** under **Storage**



Now we need to create a bucket where we want to store our photos. Choose an appropriate name for the application and a close region to you. It will have some predefined security options. Leave them as they are for now. Later we will decide the read rights on the bucket:
Please uncheck "block public access option".



We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose **Provide feedback**.

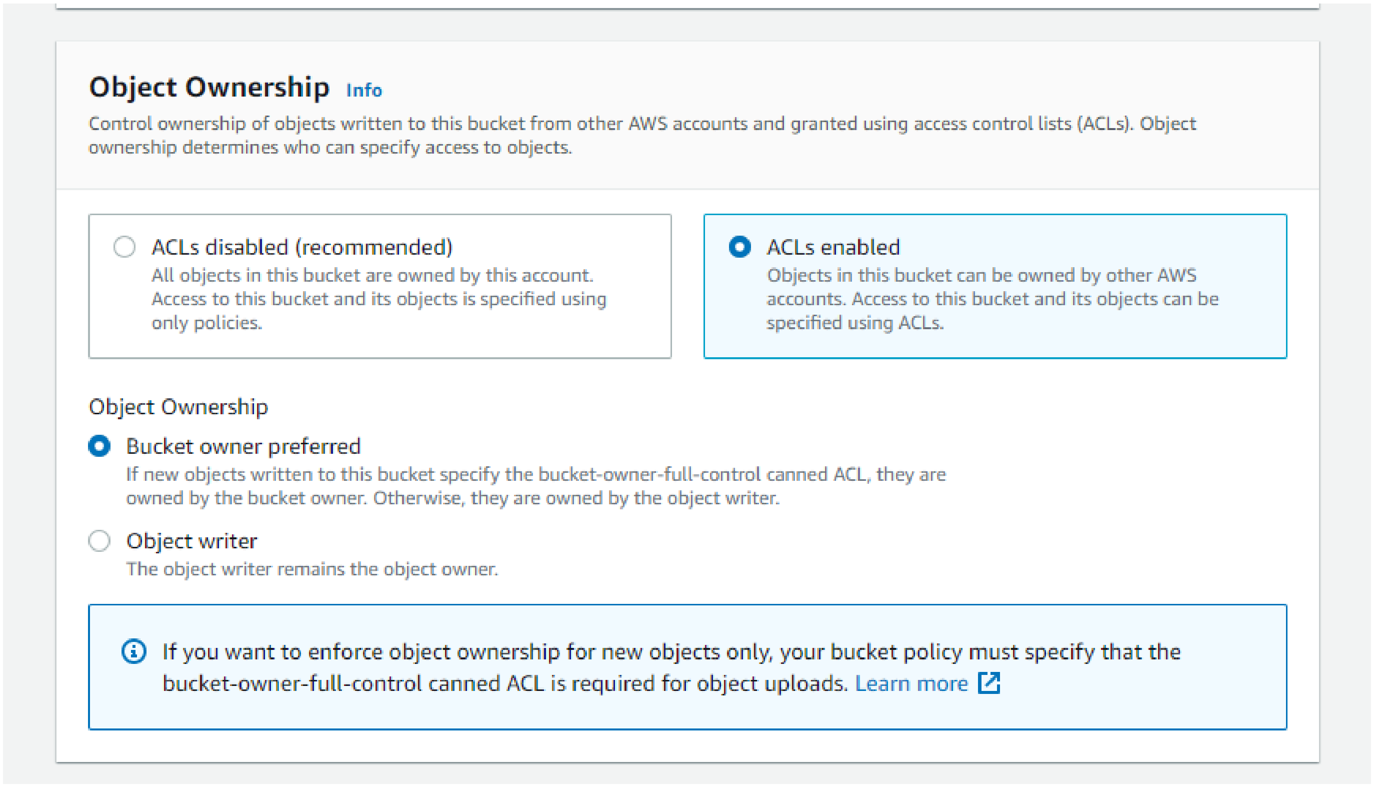
Storage lens provides visibility into storage usage and activity trends. Learn more

Buckets are containers for data stored in S3. Learn more

No buckets

You don't have any buckets.

Choose a name for your bucket and remember the region.
Then, ucheck “Block public access”.
After that select ACLs enabled option



Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and granted using access control lists (ACLs). Object ownership determines who can specify access to objects.

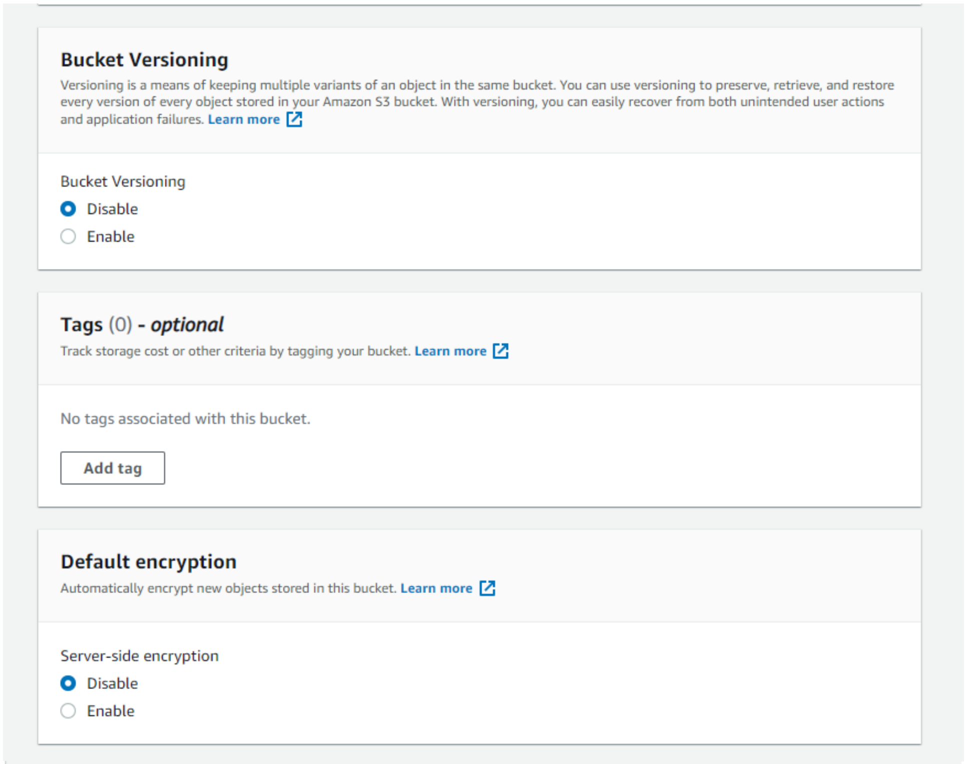
- ○ ACLs disabled (recommended)
  All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ● ACLs enabled
  Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

- ● Bucket owner preferred
  If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.
- ○ Object writer
  The object writer remains the object owner.

If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. Learn more

Other options should remain unchanged.



Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

[✓ Disable]

Enable

Tags (0) - *optional*

Track storage cost or other criteria by tagging your bucket. Learn more

No tags associated with this bucket.

Add tag

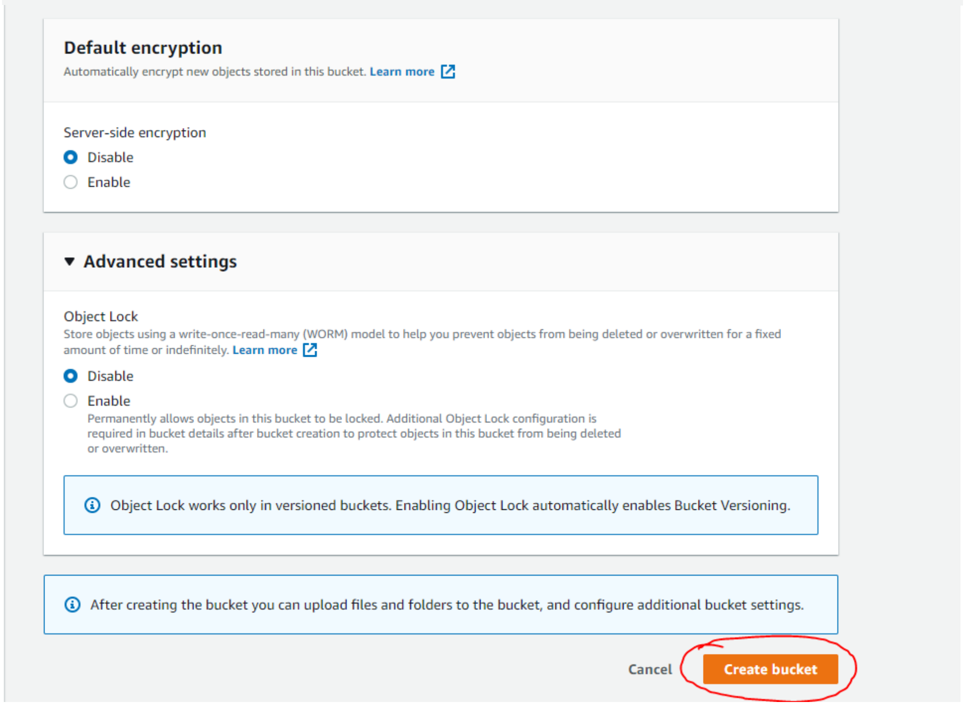
Default encryption

Automatically encrypt new objects stored in this bucket. Learn more

Server-side encryption

[✓ Disable]

Enable



Default encryption

Automatically encrypt new objects stored in this bucket. Learn more

Server-side encryption

[✓ Disable]

Enable

▼ Advanced settings

Object Lock

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Learn more

[✓ Disable]

Enable

Permanently allows objects in this bucket to be locked. Additional Object Lock configuration is required in bucket details after bucket creation to protect objects in this bucket from being deleted or overwritten.

ⓘ Object Lock works only in versioned buckets. Enabling Object Lock automatically enables Bucket Versioning.

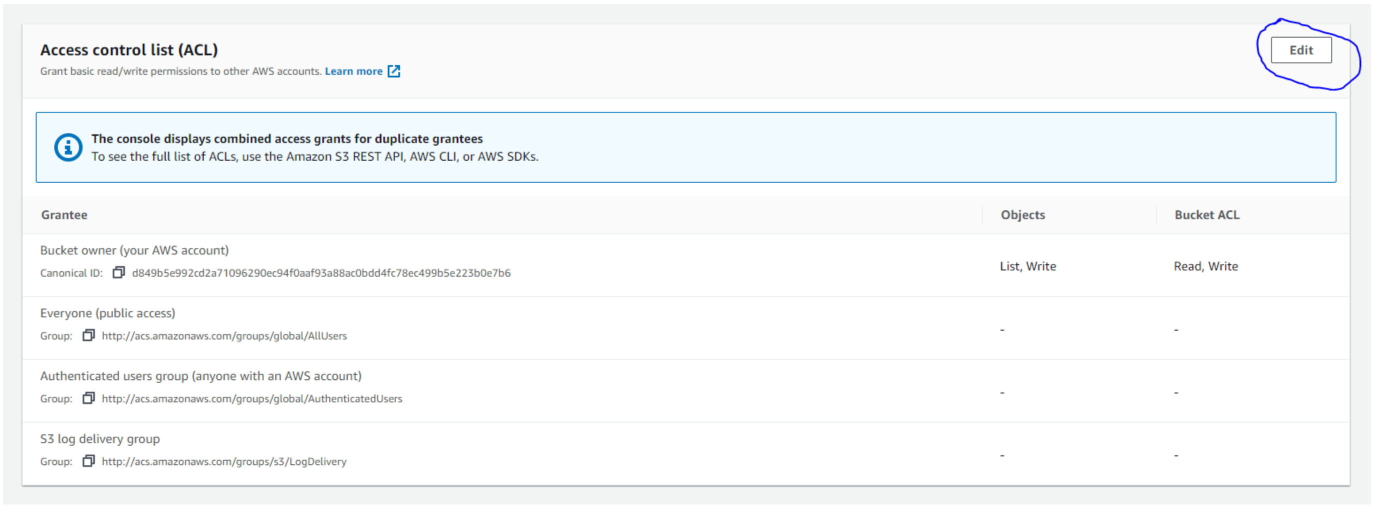
ⓘ After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

Now the bucket is created. Select your bucket and click "Persmissions", then at the"Bucket" policy section, click edit and paste the following:

```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Sid": "PublicReadGetObject",
                        "Effect": "Allow",
                        "Principal": "*",
                        "Action": "s3:GetObject",
                        "Resource": "arn:aws:s3:::complaint-system-fastapi/*"
                }
        ]
}
```

Please change the yellow part with the name of your bucket.

Click “Save changes” at the bottom right.
Scroll down to “Access control list (ACL)” and click “Edit”

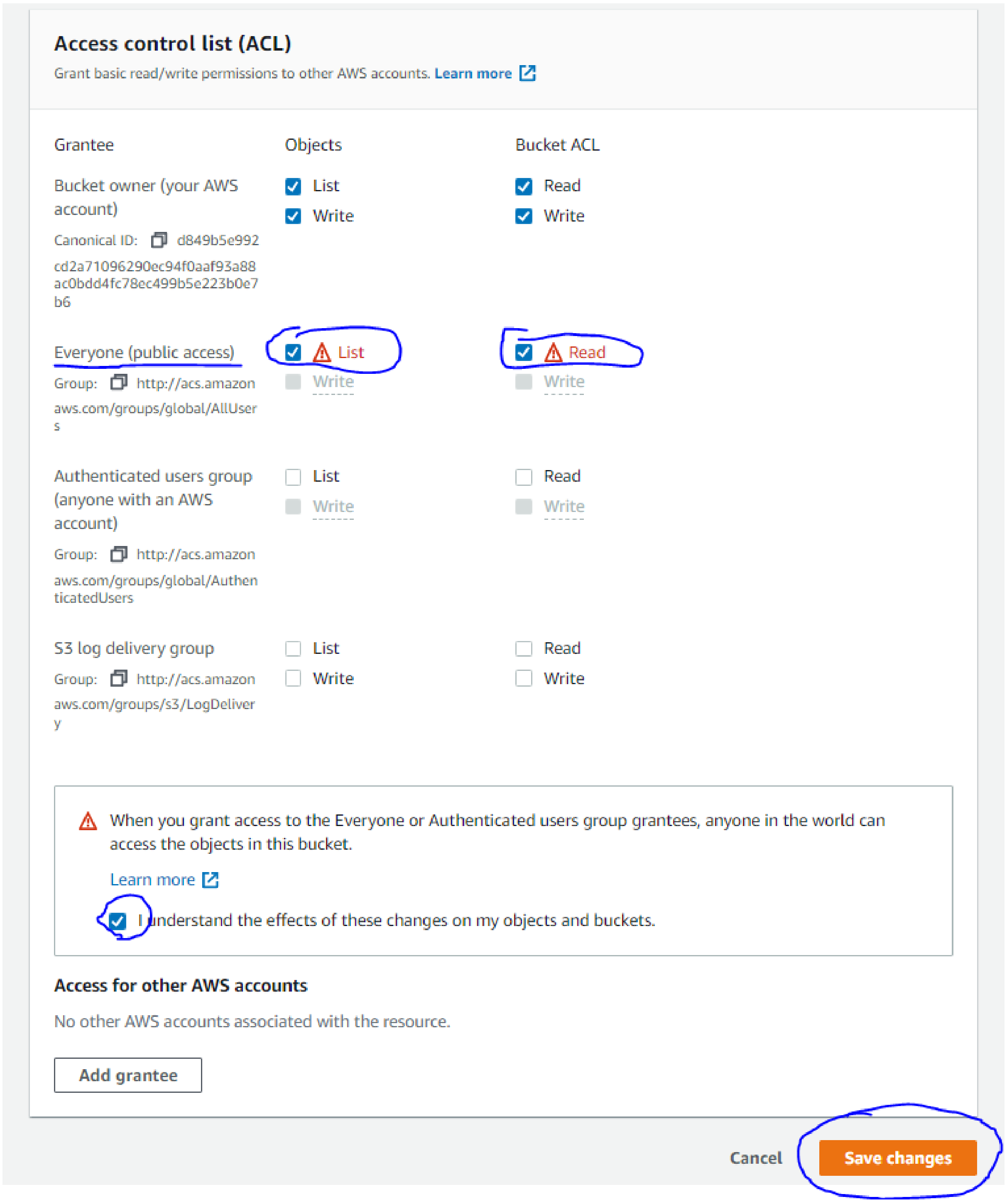


Access control list (ACL)
Grant basic read/write permissions to other AWS accounts. Learn more

The console displays combined access grants for duplicate grantees
To see the full list of ACLs, use the Amazon S3 REST API, AWS CLI, or AWS SDKs.

Grantee	Objects	Bucket ACL
Bucket owner (your AWS account) Canonical ID: d849b5e992cd2a71096290ec94f0aaf93a88ac0bdd4fc78ec499b5e223b0e7b6	List, Write	Read, Write
Everyone (public access) Group: http://acs.amazonaws.com/groups/global/AllUsers	-	-
Authenticated users group (anyone with an AWS account) Group: http://acs.amazonaws.com/groups/global/AuthenticatedUsers	-	-
S3 log delivery group Group: http://acs.amazonaws.com/groups/s3/LogDelivery	-	-

For “Everyone (public acess)” check the two options “List” and “Read”, click the ticket at the bottom and then click “Save changes”:



Access control list (ACL)

Grant basic read/write permissions to other AWS accounts. Learn more

Grantee	Objects	Bucket ACL
Bucket owner (your AWS account) Canonical ID: d849b5e992cd2a71096290ec94f0aaf93a88ac0bdd4fc78ec499b5e223b0e7b6	[✓] List [✓] Write	[✓] Read [✓] Write
Everyone (public access) Group: http://acs.amazonaws.com/groups/global/AllUsers	[✓] List [ ] Write	[✓] Read [ ] Write
Authenticated users group (anyone with an AWS account) Group: http://acs.amazonaws.com/groups/global/AuthenticatedUsers	[ ] List [ ] Write	[ ] Read [ ] Write
S3 log delivery group Group: http://acs.amazonaws.com/groups/s3/LogDelivery	[ ] List [ ] Write	[ ] Read [ ] Write

When you grant access to the Everyone or Authenticated users group grantees, anyone in the world can access the objects in this bucket.

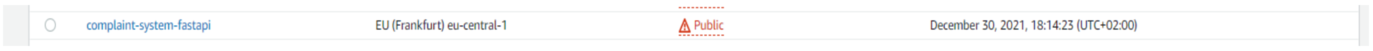
Learn more

[✓] I understand the effects of these changes on my objects and buckets.

Access for other AWS accounts

No other AWS accounts associated with the resource.

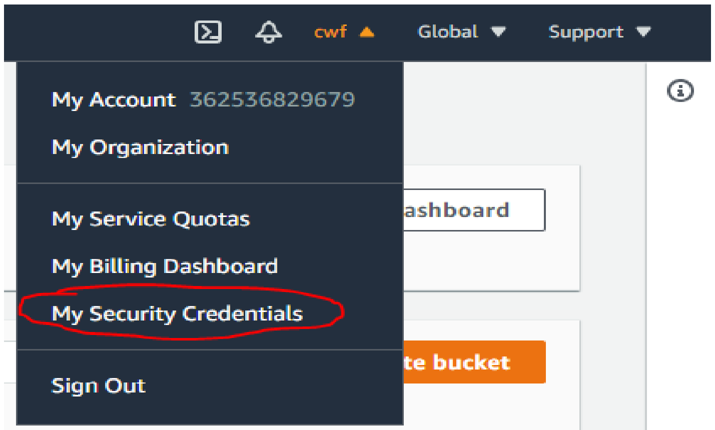
And now the bucket is public:

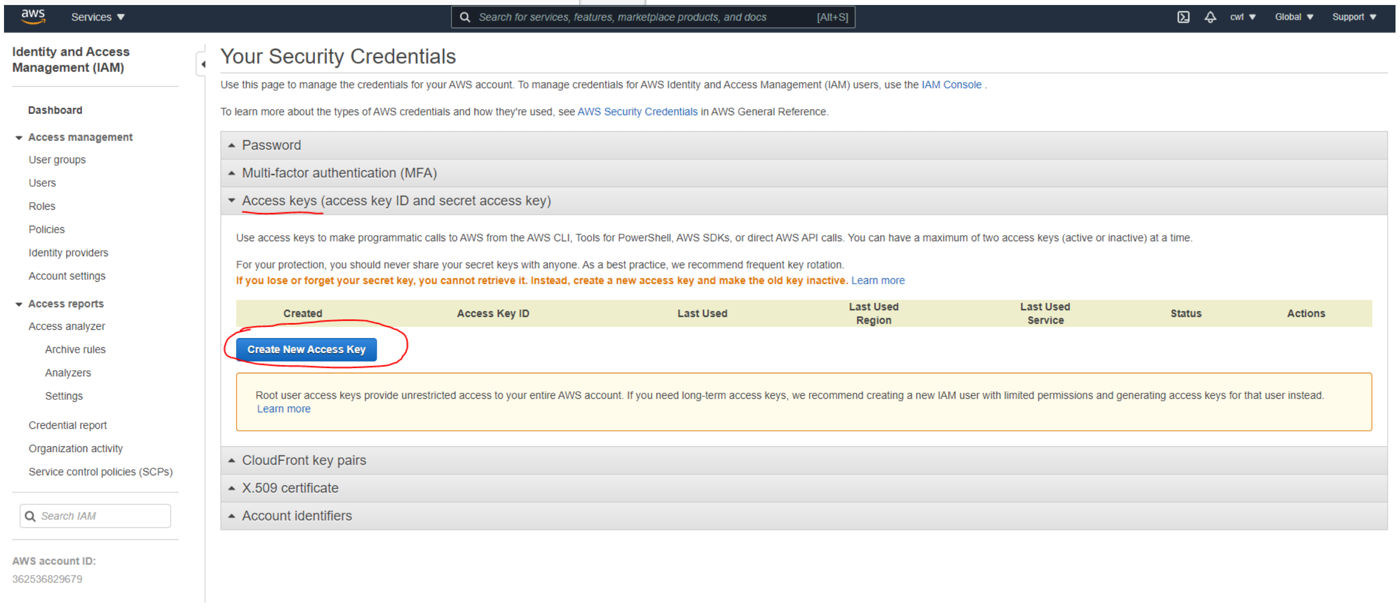


complaint-system-fastapi | EU (Frankfurt) eu-central-1 | Public | December 30, 2021, 18:14:23 (UTC+02:00)

4. Fetch credentials

To connect to AWS services with boto3, you will need to generate a key and a secret. It is essential to store these credentials right after they are generated!
Go to the upright corner on the navigation bar and click your profile name, then follow the steps:





Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the IAM Console .

To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference.

- Password
- Multi-factor authentication (MFA)
- Access keys (access key ID and secret access key)

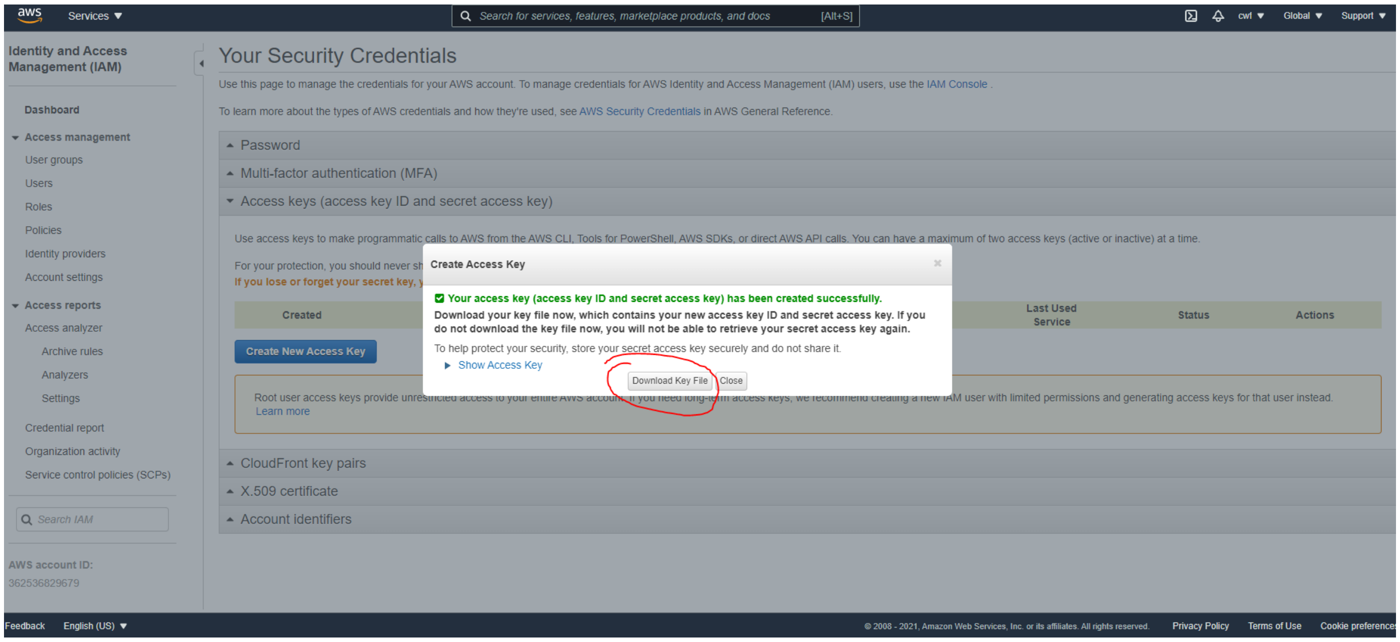
Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time.

For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation.
If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key and make the old key inactive. Learn more

Created	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions

Root user access keys provide unrestricted access to your entire AWS account. If you need long-term access keys, we recommend creating a new IAM user with limited permissions and generating access keys for that user instead. Learn more

- CloudFront key pairs
- X.509 certificate
- Account identifiers



Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the IAM Console .

To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference.

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time.

For your protection, you should never sh

If you lose or forget your secret key, y

Create Access Key

Your access key (access key ID and secret access key) has been created successfully.

Download your key file now, which contains your new access key ID and secret access key. If you do not download the key file now, you will not be able to retrieve your secret access key again.

To help protect your security, store your secret access key securely and do not share it.

Root user access keys provide unrestricted access to your entire AWS account. If you need long-term access keys, we recommend creating a new IAM user with limited permissions and generating access keys for that user instead. Learn more

AWS account ID:
362536829679

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

After you downloaded your credentials, you need to go to your **.env** file and define them:

```
AWS_ACCESS_KEY="PASTE YOUR KEY"
AWS_SECRET="PAASTE YOUR SECRET"
AWS_BUCKET="PASTE YOUR BUCKET NAME"
AWS_REGION="eu-central-1" # if you have chosen different region, the value will be
different
```

We need to make some adjustments to the current schemas. In schemas/bases.py we need to change like this:

```
class BaseComplaint(BaseModel):
    title: str
    description: str
    amount: float
```

and in the schemas/request/complaint.py:

```
from schemas.base import BaseComplaint


class ComplaintIn(BaseComplaint):
    encoded_photo: str
    extension: str
```

We will create a folder called 'temp_files' under the project's root. Its purpose will be to store the files the user has sent after we decode them. Add it to .gitignore.
Next, in the root of our project we will create a file called **constants.py**
It is really important to work with join especially if you are on windows (because of the \\ and / difference in the unix alike systems and windows).

```
import os

ROOT_DIR = os.path.dirname(os.path.abspath(__file__))
TEMP_FILE_FOLDER = os.path.join(ROOT_DIR, 'temp_files')
```

In **utils** folder create a file called **helpers.py.** Here we will define a function which will help us to decode the photo:

```
import base64

from fastapi import HTTPException


def decode_photo(path, encoded_string):
    with open(path, "wb") as f:
        try:
            f.write(base64.b64decode(encoded_string.encode("utf-8")))
        except Exception as ex:
            raise HTTPException(status_code=400, detail="Invalid photo encoding")
```

Now we will create our S3Service, responsible for the communication between our app and s3:

```
import boto3
from botocore.exceptions import ClientError


from decouple import config
from fastapi import HTTPException


class S3Service:
    def __init__(self):
        self.key = config("AWS_ACCESS_KEY")
        self.secret = config("AWS_SECRET")
        self.s3 = boto3.client(
            "s3", aws_access_key_id=self.key, aws_secret_access_key=self.secret,
        )
        self.bucket = config("AWS_BUCKET")

    def upload_photo(self, path, key, ext):
        try:
            self.s3.upload_file(path, self.bucket, key, ExtraArgs={'ACL': 'public-
read', 'ContentType': f'image/{ext}'})
            return
f"https://{config('AWS_BUCKET')}.s3.{config('AWS_REGION')}.amazonaws.com/{key}"
        except ClientError as ex:
            raise HTTPException(status_code=500, detail="S3 is not available at
the moment")
        except Exception as ex:
            raise HTTPException(status_code=500, detail="S3 is not available at
the moment")
```

In the init method we are setting up the key and the secret we obtained from the AWS console.
Then we define a function which uploads the file with the help of the s3 client from boto library and return the URL of the photo. The path is the temp_folder/file_name.extension which we have already decode and stored locally.
The key is the name and the extension of the file.

We have done so much, but now we have to update the manager to follow the newly requested functionality.
The create method of the ComplaintManager now will look like this:

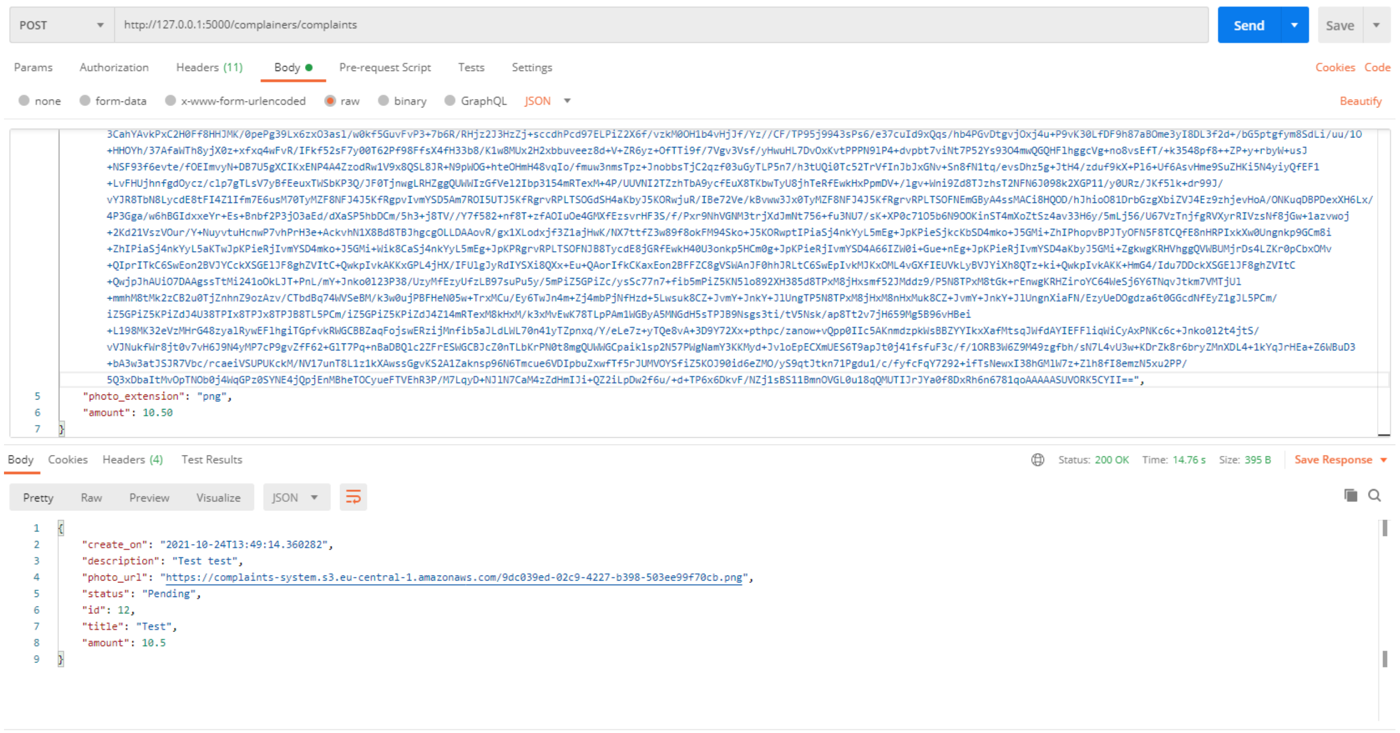
```
import os
import uuid

from constants import TEMP_FILE_FOLDER
from db import database
from models import complaint, RoleType, State, transaction
from services.s3 import S3Service
from services.wise import WiseService
from tests.ses import SESService
from utils.helpers import decode_photo


s3 = S3Service()
… (the remaining code bolow)

@staticmethod
async def create_complaint(complaint_data, user):
    data = complaint_data.dict()
    data["complainer_id"] = user["id"]
    encoded_photo = data.pop("encoded_photo")
    ext = data.pop("extension")
    name = f"{uuid.uuid4()}.{ext}"
    path = os.path.join(TEMP_FILE_FOLDER, name)
    decode_photo(path, encoded_photo)
    data["photo_url"] = s3.upload_photo(path, name, ext)
    id_ = await database.execute(complaint.insert().values(**data))
    return await database.fetch_one(complaint.select().where(complaint.c.id ==
id_))
```

We added a couple of steps – we remove the **photo** and **photo_extension** because they are not part of our model, but we store their values. We build a uuid and extension for the file name. Then we decode the photo (which will decode the file, give it this name and place it to the temp_folder). Then using this path we will upload the photo to s3 with key the name we generated and save the URL to data and then to the database.
The request response will look like this:



POST http://127.0.0.1:5000/complainers/complaints

```
3CahYAvkPxC2H0Ff8HHJMK/0pePg39Lx6zxO3asl/w0kf5GuvFvP3+7b6R/RHjz2J3HzZj+sccdhPcd97ELPiZ2X6f/vzkM0OH1b4vHjJf/Yz//CF/TP95j9943sPs6/e37cuId9xQqs/hb4PGvDtgvjOxj4u+P9vK30LfDF9h87aBOme3yI8DL3f2d+/bG5ptgfym8SdLi/uu/1O
+HHOYh/37AfaWTh8yjX0z+xfxq4wFvR/IFkf52sF7y00T62Pf98FfsX4fH33b8/K1w8MUx2H2xbbuveez8d+V+ZR6yz+OfTTi9f/7Vgv3Vsf/yHwuHL7DvOxKvtPPPN91P4+dvpbt7viNt7P52Ys93O4mwQGQHFlhggcVg+no8vsEfT/+k3548pf8++ZP+y+rbyW+usJ
+NSF93f6evte/fOEImvyN+DB7U5gXCIKxENP4A4ZzodRw1V9x8QSL8JR+N9pWOG+hteOHmH48vqIo/fmuw3nmsTpz+JnobbsTjC2qzf03uGyTLP5n7/h3tUQi0Tc52TrVfInJbJxGNv+Sn8fN1tq/evsDhz5g+JtH4/zduf9kX+P16+Uf6AsvHme9SuZHKi5N4yiyQfEF1
+LvFHUjhnfgdOycz/clp7gTLsV7yBfEeuxTWSbKP3Q/JF0TjnwgLRHZggQUWWIzGfVel2Ibp3154mRTexM+4P/UUVNI2TZzhTbA9ycfEuX8TKbwTyU8jhTeRfEwkHxPpmDV+/1gv+Wni9Zd8TJzhsT2NFN6J098k2XGP11/y0URz/JKf51k+dr99J/
vYJR8TbN8LycdE8tFI4Z1Ifm7E6usM70TyMZF8NFJ4J5KfRgpvIvmYSD5Am7ROI5UTJ5KfRgrvRPLTSOGdSH4aKbyJ5KORwjuR/IBe72Ve/kBvww3Jx0TyMZF8NFJ4J5KfRgrvRPLTSOFNEmGByA4ssMACi8HQOD/hJhioO81DrbGzgXbiZVJ4Ez9zhjevHoA/ONKuqDBPDexXH6Lx/
4P3Gga/w6hBGIdxxeYr+Es+Bnbf2P3jO3aEd/dXaSP5hbDCm/5h3+j8TV//Y7f582+nf8T+zfAOIuOe4GMXfEzsvrHF3S/f/Pxr9NhVGNM3trjXdJmNt756+fu3NU7/sK+XP0c71O5b6N9OOKinST4mXoZtSz4av33H6y/5mLj56/U67VzTnjfgRVXyrRIVzsNf8jGw+1azvwoj
+2Kd21VszVOur/Y+NuyvtuHcnwP7vhPrH3e+AckvhN1X8Bd8TBJhgcgOLLDAAovR/gx1XLodxjf3Z1ajHwK/NX7ttfZ3w89f8okFM94Sko+J5KORwptIPiaSj4nkYyL5mEg+JpKPieSjkcKbSD4mko+J5GMi+ZhIPhopvBPJTyOFN5F8TCQfE8nHRPIxkXw0Ungnkp9GCm8i
+ZhIPiaSj4nkYyL5aKTwJpKPieRjIvmYSD4mko+J5GMi+Wik8CaSj4nkYyL5mEg+JpKPRgrvRPLTSOFNJB8TycdE8jGRfEwkH40U3onkp5HCm0g+JpKPieRjIvmYSD4A66IZW0i+Gue+nEg+JpKPieRjIvmYSD4aKbyJ5GMi+ZgkwgKRHVhggQVWBUMjrDs4LZKr0pCbxOMv
+QIprITkC6SwEon2BVJYCckXSGElJF8ghZVItC+QwkpIvkAKKxGPL4jHX/IFU1gJyRdIYSXi8QXx+Eu+QAorIfkCKaxEon2BFFZC8gVSWAnJF0hhJRLtC6SwEpIvkMJKxOML4vGXfIEUVkLyBVJYiXh8QTz+ki+QwkpIvkAKK+HmG4/Idu7DDckXSGElJF8ghZVItC
+QwjpJhAUiO7DAAgssTtMi241oOkLJT+PnL/mY+Jnko0l23P38/UzyMfEzyUfzLB97suPu5y/5mPiZ5GPiZc/ysSc77n7+fib5mPiZ5KN5lo892XH385d8TPxM8jHxsmf52JMddz9/P5N8TPxM8tGk+rEnwgKRHZiroYC64WeSj6Y6TNqvJtkm7VMTjU1
+mmhM8tMk2zCB2u0TjZnhnZ9ozAzv/CTbdBq74WVSeBM/k3w0ujPBFHeN05w+TrxMCu/Ey6TwJn4m+Zj4mbPjNfHzd+5Lwsuk8CZ+JvmY+JnkY+J1UngTP5N8TPxM8jHxM8nHxMuk8CZ+JvmY+JnkY+J1UngnXiaFN/EzyUeDOgdza6t0GGcdNfEyZ1gJL5PCm/
iZ5GPiZ5KPiZdJ4U38TPIx8TPJx8TPJB8TL5PCm/iZ5GPiZ5KPiZdJ4Z14mRTexM8kHxM/k3xMvEwK78TLpPAm1WGByA5MNGdH5sTPJB9Nsgs3ti/tV5Nsk/ap8Tt2v7jH659Mg5B96vHBei
+L198MK32eVzMHrG48zyalRywEFlhgiTGpfvkRWGCBBZaqFojswERzijMnfib5aJLdLWL70n41yTZpnxq/Y/eLe7z+yTQe8vA+3D9Y72Xx+pthpc/zanow+vQpp0IIc5AKnmdzpkWsBBZYYIkxXafMtsqJWfdAYIEFFliqWiCyAxPNKc6c+Jnko0l2t4jtS/
vVJNukfWr8jt0v7vH6J9N4yMP7cP9gvZfF62+G1T7Pq+nBaDBQ1c2ZFrESWGCBJcZ0nTLbKrPN0t8mgQUWWGCpaiklsp2N57PWgNamY3KKMyd+JvloEpECXmUES6T9apJt0j41fsfuF3c/f/1ORB3W6Z9M49zgfbh/sN7L4vU3w+KDrZk8r6bryZMnXDL4+1kYqJrHEa+Z6WBuD3
+bA3w3atJSJR7Vbc/rcaeiVSUPUKckM/NV17unT8L1z1kXAwssGgvKS2A1Zaknsp96N6Tmcue6VDIpbuZxwfTf5rJUMVOYSfiZ5KOJ90id6eZMO/yS9qtJtkn71Pgdu1/c/fyfcFqY7292+ifTsNewxI38hGM1W7z+Z1h8fI8emzN5xu2PP/
5Q3xDbaItMvOpTNOb0j4WqGPz0SYNE4jQpjEnMBheTOCyueFTVEhR3P/M7LqyD+NJ1N7CaM4zZdHmIJi+QZ2iLpDw2f6u/+d+TP6x6DkvF/NZj1sBS11BmnOVGL0u18qQMUTIJrJYa0f8DxRh6n6781qoAAAAASUVORK5CYII==",
```

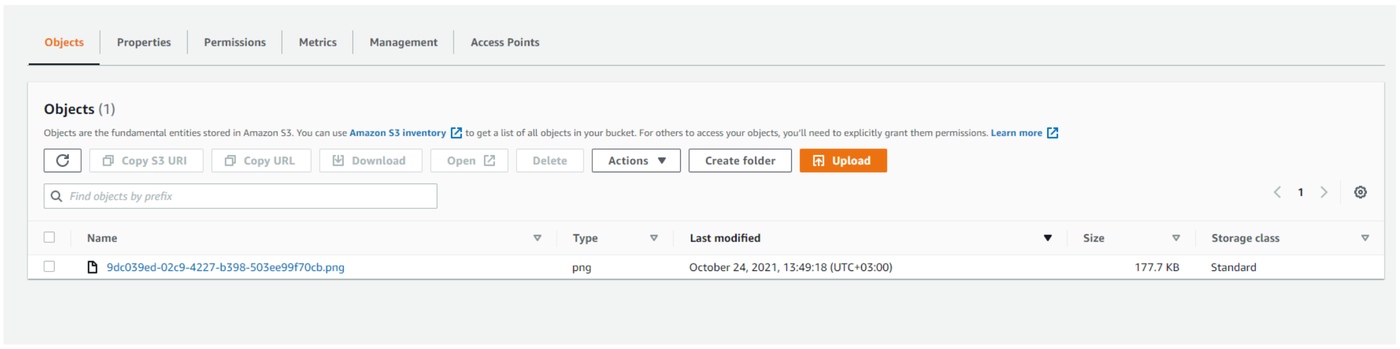
```
    "photo_extension": "png",
    "amount": 10.50
}
```

Status: 200 OK Time: 14.76 s Size: 395 B

```
{
    "create_on": "2021-10-24T13:49:14.360282",
    "description": "Test test",
    "photo_url": "https://complaints-system.s3.eu-central-1.amazonaws.com/9dc039ed-02c9-4227-b398-503ee99f70cb.png",
    "status": "Pending",
    "id": 12,
    "title": "Test",
    "amount": 10.5
}
```

(or you can try it directly on Swagger)
You can try encode a string with this online tool.

You can validate the file is uploaded by checking the bucket's content in the AWS Console.



Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
9dc039ed-02c9-4227-b398-503ee99f70cb.png	png	October 24, 2021, 13:49:18 (UTC+03:00)	177.7 KB	Standard

You should delete the locally stored photo after it is uploaded:
os.remove(path)