How to use AWS S3 Bucket to store static and media files in Django

If you're using the Django web app, you need to efficiently handle static content and user-uploaded media. AWS S3 Bucket will make it easy for you to do that in Django.

Picture 1 of How to use AWS S3 Bucket to store static and media files in Django

Amazon Web Services (AWS) Simple Storage Service (S3) Bucket is an alternative for storing static and multimedia files. By integrating S3 with Django, you can offload the file management burden on your server, offloading, and ensuring faster, more reliable content delivery.

Step 1: Create an AWS account

If you don't have an account, go to the AWS site to create a new account.

New AWS accounts get free access to 5GB of S3 standard storage per month for one year.

Step 2: Create S3 Bucket for the project

1. After creating an AWS account, log in and find S3 in the search bar at the top, then click the first option.
2. Then you will see a new page, click **Create bucket** button .
3. Next, name your **S3 bucket .** You can keep the default configuration.
4. Scroll down to **Block Public Access settings for this bucket** , uncheck **Block all public access** and confirm the warning that appears.
5. Once done, click the **Create bucket** button . It will take you to a page showing a list of the S3 buckets that have been created.

Picture 2 of How to use AWS S3 Bucket to store static and media files in Django

Step 3: Create IAM User on AWS

AWS provides a service called IAM (Identity and Access Management). This allows you to create a separate account for specific people or applications that need to interact with AWS services.

You can assign different levels of permissions to IAM users who represent individuals or applications that interact with created AWS services. With IAM users, you can ensure each user only has access to the resources they need and no more.

For security purposes, you should create an IAM user for your Django project to interact with the S3 bucket. Follow these steps to create an IAM user on AWS:

1. In the search bar, type IAM and click the first option. A new page appears:
2. On the left hand side of the IAM page, select Users, then continue to click the **Add users** button . It will open another page to fill in the information.
3. Start by entering a name for the IAM user and click the Next button at the bottom.
4. On the next page, you have to select the permission levels for the IAM user. Follow these steps:
    1. First, select **Attach policies directly** from **Permissions options** .
    2. Next, define the authorization policy for the IAM user. This action determines what an IAM user can and cannot do. Since you want the Django app to download and upload files, you should give it full access to the S3 bucket.
    3. In the **Permissions policies** section , you should find & select **S3FullAccess** . Then, click **Next** .
5. Next, review the policies for the IAM user and click the **Create user** button to create the IAM user.

Picture 3 of How to use AWS S3 Bucket to store static and media files in Django

Step 4: Generate an access key for the IAM . user

In AWS, a credential-only access key you can use to programmatically authenticate and access AWS resources. Your Django project must provide these credentials to access the S3 bucket.

The following steps will help you create an access key for the project:

1. After creating the IAM user, you will receive a prompt to view user. You can view the user by clicking User name.
2. Next, select the **Security credentials** tab > scroll down to find **Access keys** and select **Create access key** .
3. You need to choose a use case for the access, so that AWS can suggest suitable alternatives. It does not affect the access key. Feel free to click an option like **Hird-party service** or **Local code** and confirm the warning. Then, click the **Next** button .
4. On the next page, enter a description tag for the access key and click the **Create access key** button .
5. After generating the access key, you can copy the credentials or download them as a CSV file. Either way, make sure this data is safe and secure.

Picture 4 of How to use AWS S3 Bucket to store static and media files in Django

Step 5: Configure Django project for S3 Bucket

To use S3 buckets with Django projects, install the following packages:

1. **django-storages** helps you define a storage backend for files.

2. **boto3** is the AWS Software Development Kit (SDK) that helps Python projects interact with AWS.

You can install these packages into a virtual Python environment with Python's Pip package manager with the following command in the terminal:

```
pip install django-storages boto3
```

After successfully installing these packages, open the **settings.py** file and add **boto3** to the installed app.

Finally, configure the Django project to use the AWS S3 bucket. Here is the common configuration:

```
AWS_ACCESS_KEY_ID = 'AWS_ACCESS_KEY_ID ' AWS_SECRET_ACCESS_KEY = 'AWS_SECRET_ACCI
```

Paste the above configuration into the **settings.py** file and change the value accordingly. Replace **AWS_ACCESS_KEY_ID** and **AWS_SECRET_ACCESS_KEY** with the access key and secret access key you originally copied or downloaded. You should also change **AWS_STORAGE_BUCKET_NAME and AWS_S3_REGION_NAME** to the name of the bucket and the S3 bucket.

You can get the domain name by navigating to the S3 bucket and copying the last value in the **AWS region** column .

Picture 5 of How to use AWS S3 Bucket to store static and media files in Django

Step 6: Check AWS Configuration

The following code sample will upload files directly from the admin panel, but you are free to upload data from other locations.

Here, you have a sample like this:

```
class Post(models.Model): title = models.CharField(max_length=225, blank=False, r
```

Make sure you implement the necessary operations like migration, adding it to the admin panel, creating a view window and other things needed for the project. Make sure you follow Django's MVT rules.

Once done, navigate to the admin panel or the created form for file upload and upload an image file.

Navigate to the main page and claim the photo there. If so, right-click the image, then select **Open image in new tab** . In the new tab containing the images, you'll see an address bar that references the S3 bucket you created from scratch:

Picture 6 of How to use AWS S3 Bucket to store static and media files in Django

Step 7: Collect static files for bucket S3

Add the following configurations to the **settings.py** file :

```
STATICFILES_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage' AWS_LOCATION =
```

Then, open the Command Line Interface (CLI) and run the command:

```
python manage.py collectstatic --noinput
```

To confirm everything is working properly, open the S3 bucket in the AWS console. You should see a folder named **static** .

It's done! Hope the article is useful to you.