# Django S3 Setup and Deployment Guide

## Prerequisites

### AWS Account Setup

1. Create an AWS account if you haven’t already
2. Install AWS CLI and configure with your credentials
3. Create an S3 bucket for your Django media files
4. Configure Security Groups:
   * Option 1: Using the AWS Console
     1. Navigate to EC2 > Security Groups
     2. Create a new security group named “django-app-sg”
     3. Add inbound rules:
        + HTTP (Port 80) from 0.0.0.0/0
        + HTTPS (Port 443) from 0.0.0.0/0
        + Custom TCP (Port 8000) from 0.0.0.0/0
        + SSH (Port 22) from your IP address only
   * Option 2: Using AWS CLI
   * # Create security group  
     aws ec2 create-security-group --group-name django-app-sg --description "Security group for Django application"  
       
     # Add inbound rules  
     aws ec2 authorize-security-group-ingress --group-name django-app-sg --protocol tcp --port 80 --cidr 0.0.0.0/0  
     aws ec2 authorize-security-group-ingress --group-name django-app-sg --protocol tcp --port 443 --cidr 0.0.0.0/0  
     aws ec2 authorize-security-group-ingress --group-name django-app-sg --protocol tcp --port 8000 --cidr 0.0.0.0/0  
     aws ec2 authorize-security-group-ingress --group-name django-app-sg --protocol tcp --port 22 --cidr $(curl -s ifconfig.me)/32

### Install Docker and Docker Compose

# SSH into your EC2 instance  
ssh -t -i blog\_project.pem ubuntu@YOUR\_EC2\_IP "sudo apt update && sudo apt install docker-compose -y"  
  
# Verify installation  
docker-compose --version

### Required Python Packages

pip install django-storages boto3

## Step 1: Configure Django Settings

Add the following to your settings.py:

INSTALLED\_APPS = [  
 ...  
 'storages',  
]  
  
# S3 Configuration  
AWS\_ACCESS\_KEY\_ID = os.environ.get('AWS\_ACCESS\_KEY\_ID')  
AWS\_SECRET\_ACCESS\_KEY = os.environ.get('AWS\_SECRET\_ACCESS\_KEY')  
AWS\_STORAGE\_BUCKET\_NAME = 'your-bucket-name'  
AWS\_S3\_REGION\_NAME = 'your-region'  
AWS\_S3\_FILE\_OVERWRITE = False  
AWS\_DEFAULT\_ACL = None  
DEFAULT\_FILE\_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'

## Step 2: Set Environment Variables

Create a .env file on your EC2 instance (NOT on your local machine):

# SSH into your EC2 instance first  
ssh -i blog\_project.pem ubuntu@YOUR\_EC2\_IP  
  
# Create and edit the .env file  
cat > .env << EOF  
AWS\_ACCESS\_KEY\_ID=your\_access\_key  
AWS\_SECRET\_ACCESS\_KEY=your\_secret\_key  
EOF  
  
# Set proper permissions  
chmod 600 .env

## Step 3: Configure CORS for S3 Bucket

Configure CORS in AWS S3 Console: 1. Go to AWS S3 Console 2. Select your bucket 3. Click on “Permissions” tab 4. Scroll down to “Cross-origin resource sharing (CORS)” 5. Click “Edit” 6. Add the following configuration:

{  
 "CORSRules": [  
 {  
 "AllowedOrigins": ["\*"],  
 "AllowedHeaders": ["\*"],  
 "AllowedMethods": ["GET", "POST", "PUT", "DELETE"],  
 "MaxAgeSeconds": 3000  
 }  
 ]  
}

Note: This CORS configuration is separate from bucket policies. If you already have a CORS configuration, this will replace it. For existing bucket policies, this CORS configuration should be added as a separate setting.

## Step 4: Deploy Container with AWS Credentials

# Get the current Git commit SHA  
GIT\_COMMIT=$(git rev-parse --short HEAD)  
  
# Build and tag the Docker image with the commit SHA  
docker build -t django-app:${GIT\_COMMIT} .  
  
# Create a deployment script that includes AWS credentials  
cat > deploy.sh << EOF  
#!/bin/bash  
docker run -d \  
 -e AWS\_ACCESS\_KEY\_ID=\${AWS\_ACCESS\_KEY\_ID} \  
 -e AWS\_SECRET\_ACCESS\_KEY=\${AWS\_SECRET\_ACCESS\_KEY} \  
 -e GIT\_COMMIT=\${GIT\_COMMIT} \  
 -p 8000:8000 \  
 django-app:\${GIT\_COMMIT}  
EOF  
  
chmod +x deploy.sh  
./deploy.sh

## Step 5: Test File Upload

1. Create a test view in your Django application
2. Upload a file through your application
3. Verify the file appears in your S3 bucket

## Security Considerations

1. Never commit .env files to version control
2. Use IAM roles when possible instead of access keys
3. Regularly rotate AWS credentials
4. Keep your security group rules as restrictive as possible
5. Monitor S3 bucket access logs
6. Enable S3 bucket versioning for file history

## Troubleshooting

1. Check AWS credentials are correctly set
2. Verify S3 bucket permissions
3. Review security group configurations
4. Check Django storage settings
5. Monitor application logs for S3 errors