Deploying to AWS: EC2 for Django Backend and S3 for React Frontend

Prerequisites:

- A Django project that works locally.
- AWS account.

Django Backend Setup

1. **Install Required Packages**:

makefile

Copy code

```
asgiref==3.8.1

Django==5.0.7

django-cors-headers==4.4.0

djangorestframework==3.15.2

djangorestframework-simplejwt==5.3.1

PyJWT==2.8.0

sqlparse==0.5.1

tzdata==2024.1
```

2. **Configure settings.py**:

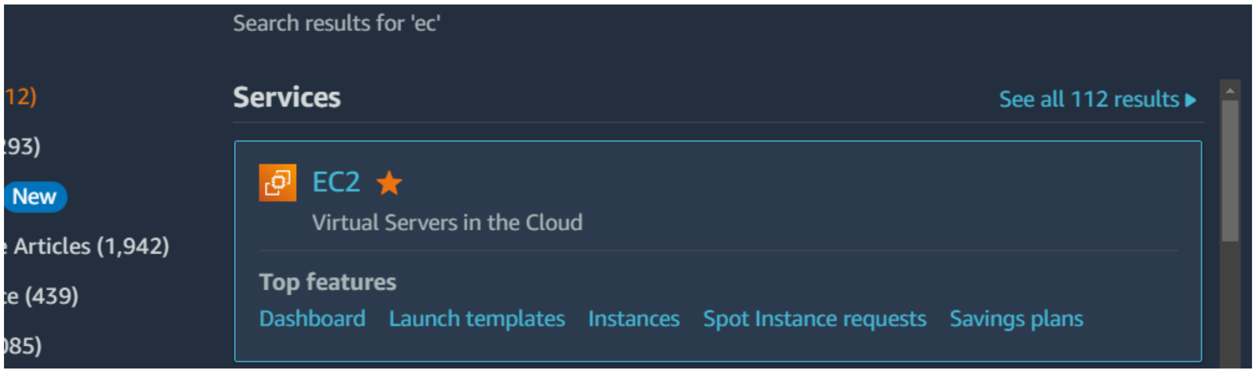
python

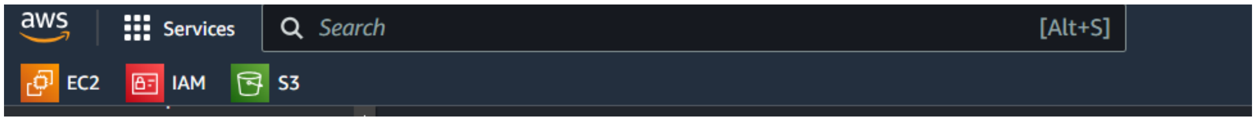
Copy code

```
INSTALLED_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'rest_framework',

    'app_your_app_name',

    'corsheaders',
    'rest_framework_simplejwt.token_blacklist'
]
```

Initial AWS Setup

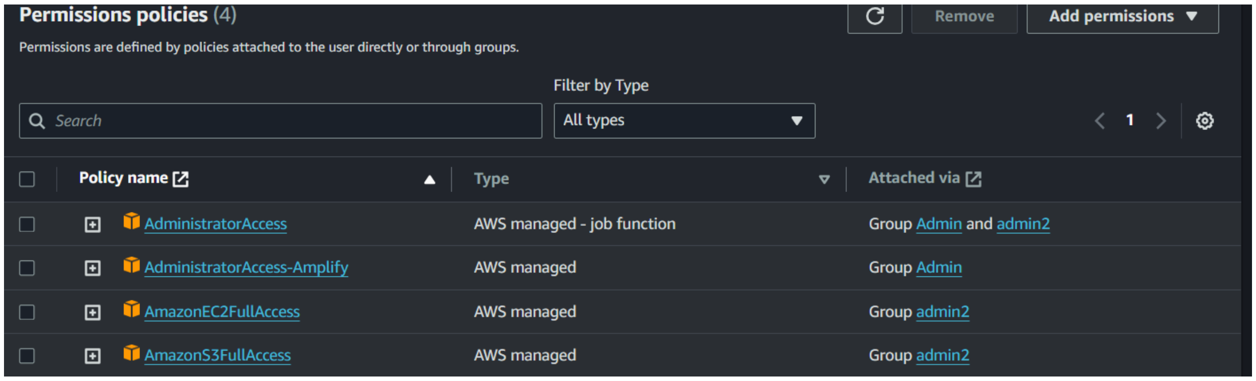
1. **Create AWS Root User**:
   - Note your account number.
   - Use the root user only for admin account overrides.
2. **AWS Console Favorites**:
   - Search for and favorite EC2, IAM, and S3.





IAM User and Group Setup

1. **Create User Groups**:
   - Create an admin group with EC2FullAccess and S3FullAccess.
   - Create other groups as needed.



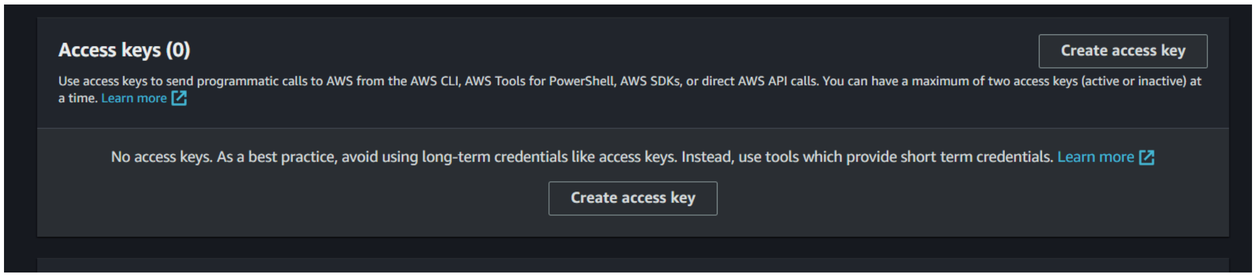
Permissions policies (4)

Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

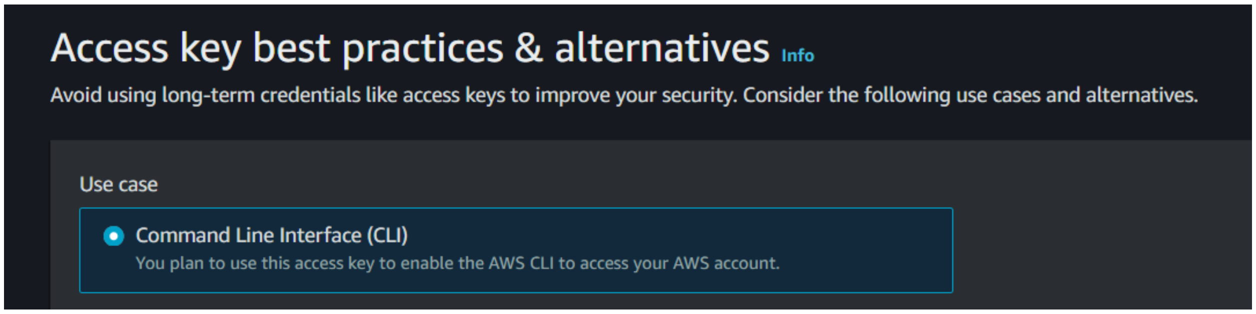
Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Group Admin and admin2
AdministratorAccess-Amplify	AWS managed	Group Admin
AmazonEC2FullAccess	AWS managed	Group admin2
AmazonS3FullAccess	AWS managed	Group admin2

2. **Create Users**:
    - Add users to the appropriate groups.
    - Generate access keys for these users.
3. **Generate AWS CLI Key**:
    - Create an access key and secret access key.


Access keys (0)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS 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. Learn more

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more


Access key best practices & alternatives Info

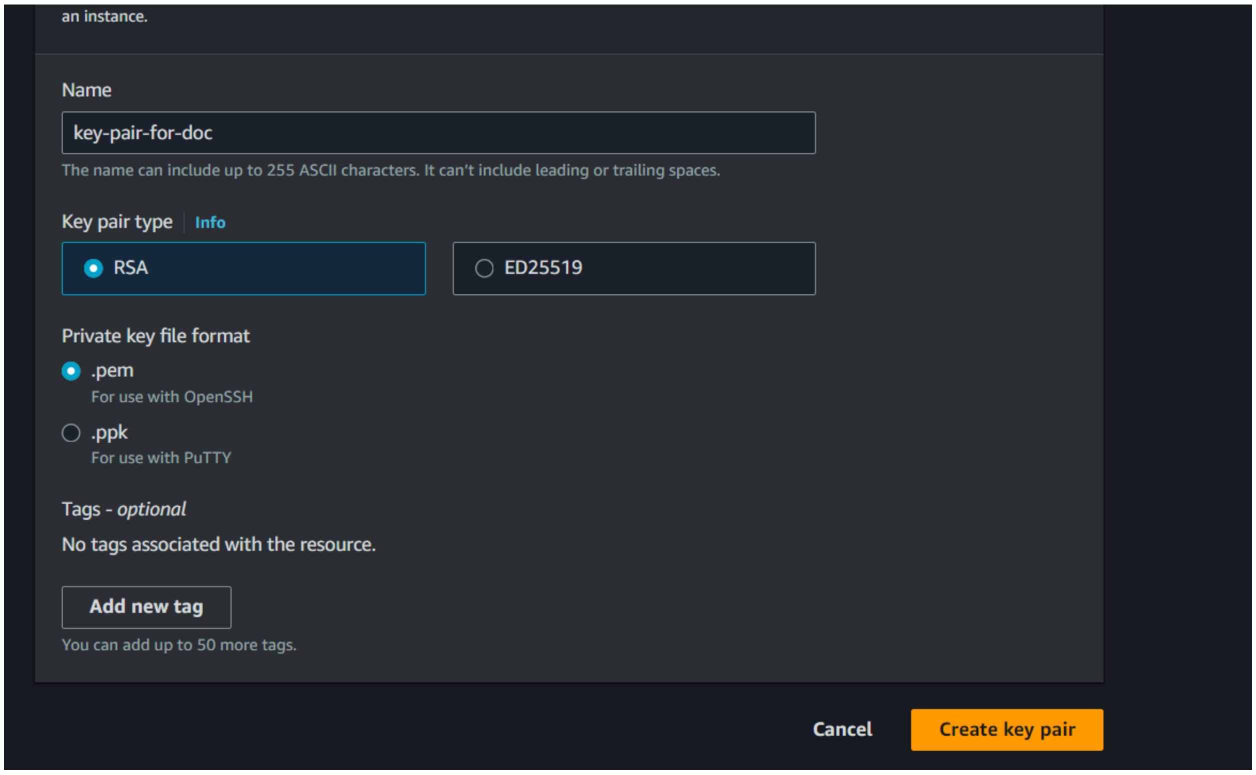
Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

● Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

EC2 Setup

1. **Create Key Pair**:
    - Generate a key pair, download the .pem file, and store it securely.
    - Modify permissions:
        - Disable inheritance.
        - Remove all users except your own.
    - Check permissions:


an instance.

Name

key-pair-for-doc

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

[✓ RSA] ED25519

Private key file format

[✓ .pem] For use with OpenSSH

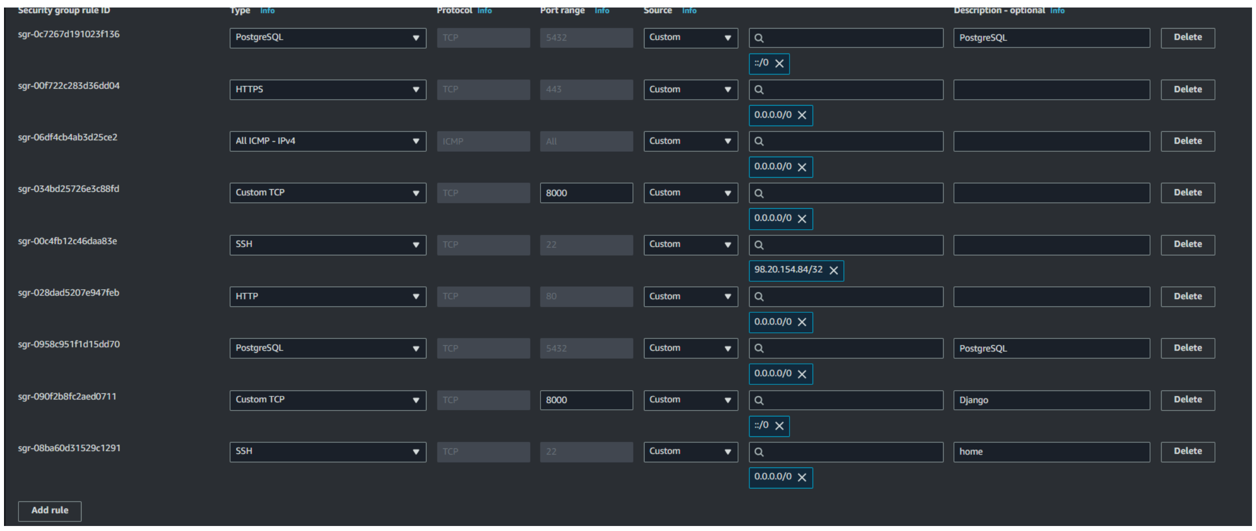
.ppk For use with PuTTY

Tags - *optional*

No tags associated with the resource.

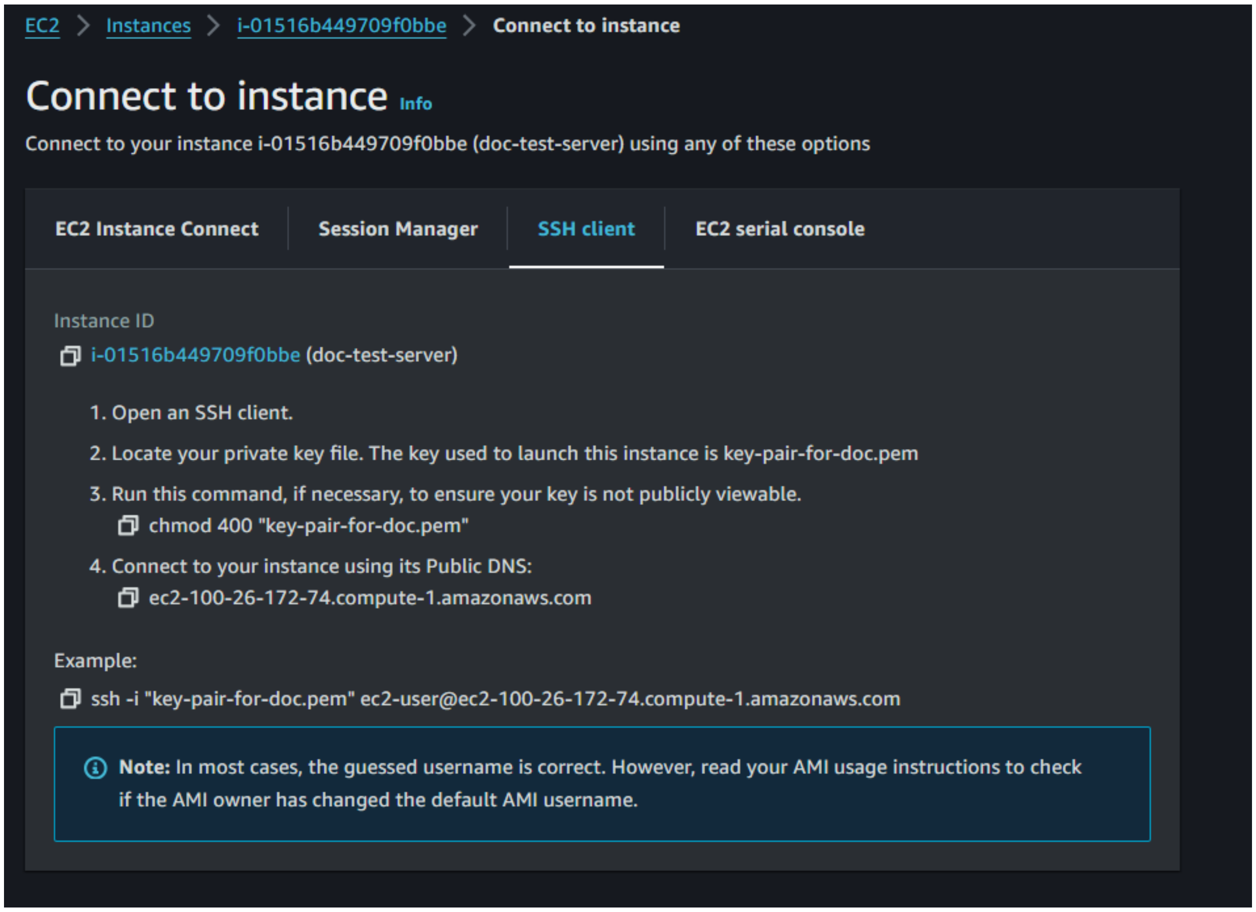
You can add up to 50 more tags.

2. **Create Security Group**:
   - Allow necessary inbound rules:
     - PostgreSQL: ::/0
     - HTTPS: 0.0.0.0/0
     - ICMP - IPv4: 0.0.0.0/0
     - Custom TCP: 0.0.0.0/0
     - SSH: 98.20.154.84/32 (or your IP)
     - HTTP: 0.0.0.0/0



Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-0c7267d191023f136	PostgreSQL	TCP	5432	Custom	::/0	PostgreSQL
sgr-00f722c283d36dd04	HTTPS	TCP	443	Custom	0.0.0.0/0	
sgr-06df4cb4ab3d25ce2	All ICMP - IPv4	ICMP	All	Custom	0.0.0.0/0	
sgr-034bd25726e3c88fd	Custom TCP	TCP	8000	Custom	0.0.0.0/0	
sgr-00c4fb12c46daa83e	SSH	TCP	22	Custom	98.20.154.84/32	
sgr-028dad5207e947feb	HTTP	TCP	80	Custom	0.0.0.0/0	
sgr-0958c951f1d15dd70	PostgreSQL	TCP	5432	Custom	0.0.0.0/0	PostgreSQL
sgr-090f2b8fc2aed0711	Custom TCP	TCP	8000	Custom	::/0	Django
sgr-08ba60d31529c1291	SSH	TCP	22	Custom	0.0.0.0/0	home

3. **Launch EC2 Instance**:
    - Use Ubuntu and t2.micro instance.
    - Select your key-pair and security group.
    - Start the instance and SSH into it. Use the SSH command shown in the EC2 console.
    - Find the SSH info by selecting connect in instances



Connect to instance Info

Connect to your instance i-01516b449709f0bbe (doc-test-server) using any of these options

Instance ID

i-01516b449709f0bbe (doc-test-server)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is key-pair-for-doc.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 "key-pair-for-doc.pem"
```

4. Connect to your instance using its Public DNS:

```
ec2-100-26-172-74.compute-1.amazonaws.com
```

Example:

```
ssh -i "key-pair-for-doc.pem" ec2-user@ec2-100-26-172-74.compute-1.amazonaws.com
```

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Step-by-Step Guide to Set Up Your Project from GitHub

1. **Install Git**:

```
sudo apt update
sudo apt install git -y
```

2. **Clone Your GitHub Repository**:

```
git clone https://github.com/your-username/your-repo.git
cd your-repo
```

3. **Set Up a Virtual Environment**:

```
python3 -m venv venv
source venv/bin/activate
```

4. **Install Project Dependencies**:

```
pip install -r requirements.txt
pip install gunicorn
```

5. **Apply Migrations**:

```
python manage.py migrate
```

6. **Install Python and Dependencies**:

```
sudo apt update
sudo apt install python3 python3-pip -y
sudo apt install python3-venv python3-dev python3-virtualenv -y
```

7. **Verify Installation**:

```
python3 --version
pip3 --version
django-admin --version
```

8. **Run the Development Server**:

```
python manage.py runserver 0.0.0.0:8000
```

   - Ensure no firewall rules on the EC2 instance itself block port 8000.
   - Check UFW Status:

```
sudo ufw status
sudo ufw allow 8000
```

9. **Use nohup to Keep the Server Running**:

```
nohup python manage.py runserver 0.0.0.0:8000 &
```

S3 Frontend Setup

1. **Create an S3 Bucket**:
   - Navigate to S3 from the AWS GUI.
   - Create a bucket.
   - Uncheck the "Block all public access" option.
   - Update bucket policy:

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

2. **Configure React App**:
   - Ensure the .env file in your React app points to the IP of the server.
   - Add the frontend to the allowed hosts in your Django backend CORS settings.
   - Run the build command:

```
npm run build
```

3. **Deploy Frontend to S3**:
   - Install AWS CLI on Windows using the official guide.
   - Configure AWS CLI:

```
aws configure
```

   - Use the following command to copy the dist folder to your S3 bucket:

```
aws s3 sync dist/ s3://your-bucket-name/ --delete
```