

How to deploy Django Application in AWS ECS

Introduction to Django

Django is a python framework for developing and building web application in the python. It supports rapid development and scalability, It follows MVS architecture for developing web apps.

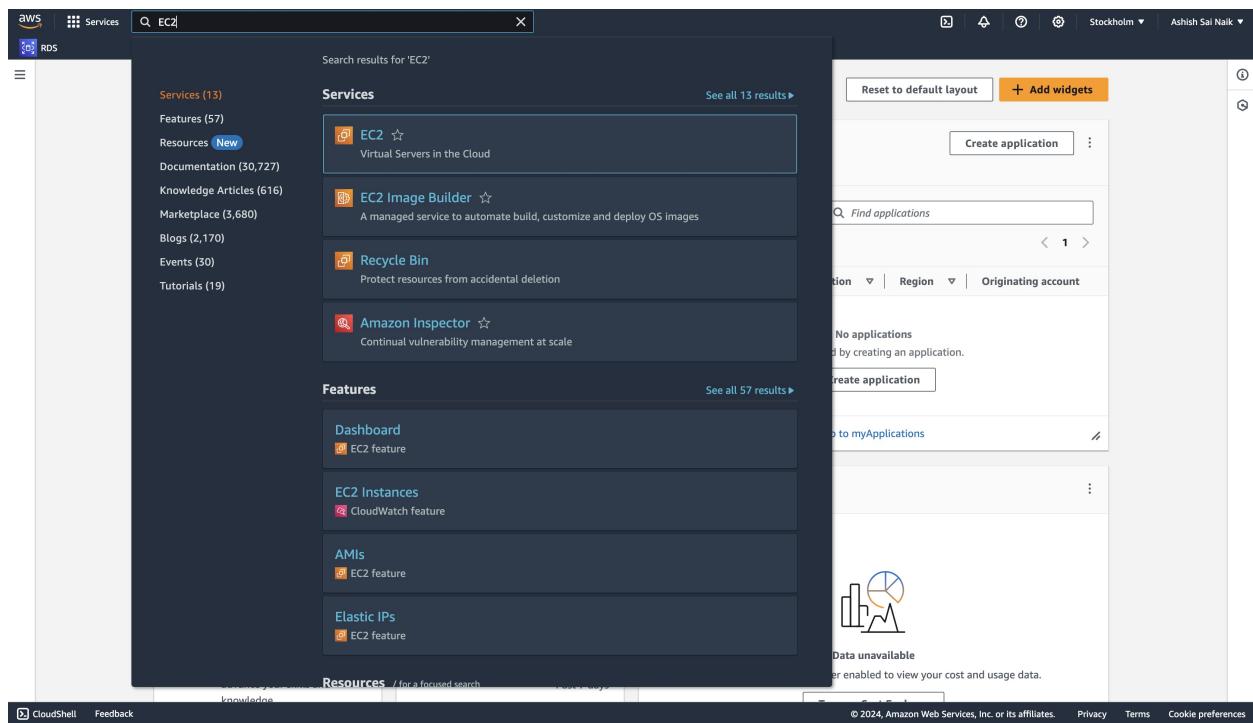
Creating EC2 Instance

Prerequisites

- An active AWS account.
- AWS Management Console login credentials (username and password, or access to an IAM user with the necessary permissions).

1. Login your AWS account and open EC2 Console

once logged in , you will be on the AWS Management Console home page. from the home page, locate the "search" at the top of the home page, In the "Search" , you can type EC2 , click on "EC2"



Once you Clicked on the EC2, You will taken to the EC2 Dashboard,

- Here, you can manage your EC2 instances, create new instances, configure security groups, manage elastic IPs, and perform other EC2-related tasks.

2. Click Launch Instance

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with various navigation options like Instances, Images, and Network & Security. The main area displays resource statistics (Instances running: 1, Auto Scaling Groups: 0, Dedicated Hosts: 0, etc.) and a prominent "Launch instance" button. A red box highlights this button, and a black arrow points to it from the left sidebar.

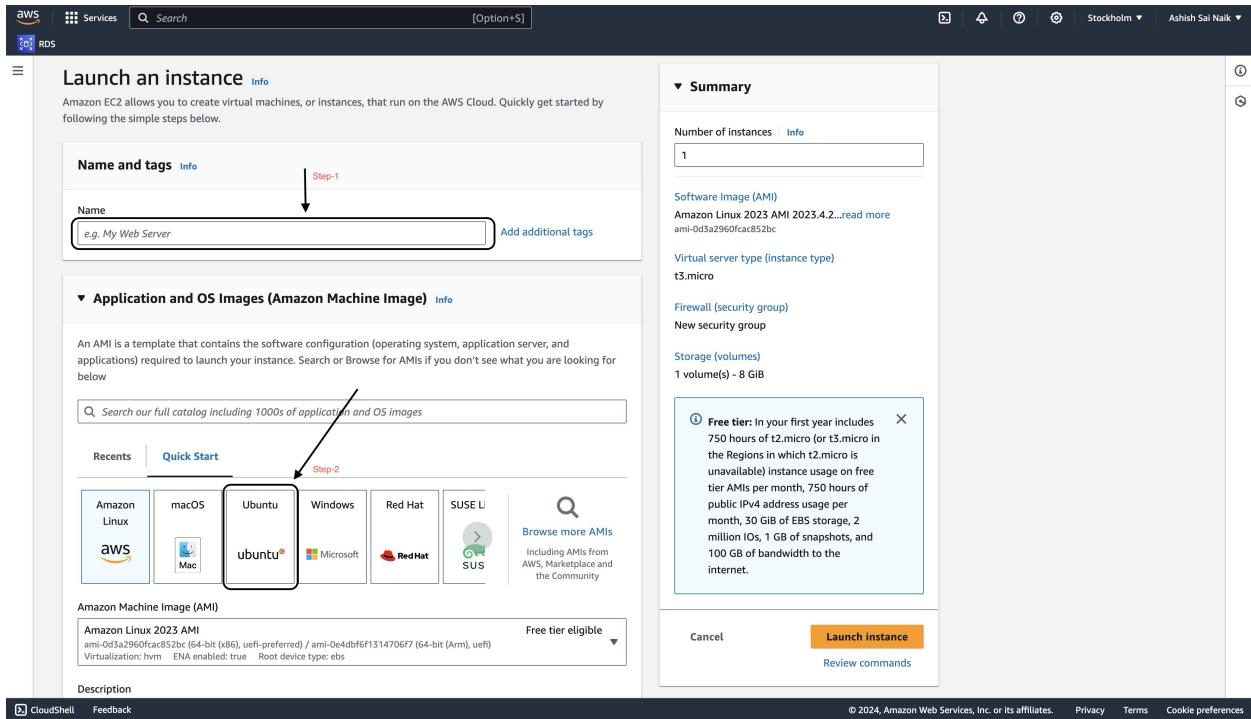
- In the EC2 Dashboard, click on the "Launch Instance" button.
- Follow the steps in the instance launch wizard to configure and launch your new instance.

Step-1

- Once you entered in the Launch an instance, You have to Enter the Name and tag for the Instance

Step-2

- Then once you completed the step-1 make sure opt the Ubuntu for the application and os image as shown in the below fig.



Step-3

- Then in the Instance type you have to opt the t3.micro , its for free tire eligible

step-4

- Then create the Key pair

Purpose

A key pair consists of a public key and a private key. It is used to securely SSH (Secure Shell) into your EC2 instances without the need for a password. This enhances security and simplifies the login process.

Step-3

Step-4

Step-4

Then in the Network Settings make sure the enable Allow SHH traffic From "Anywhere" so we can access the website

Finally we are ready to launch instance , which present in the right corner of the website

Navigate to Instances then check status of running instance

After successfully Launching the instance , return to the EC2 Dashboard Navigate to instance (running)

The screenshot shows the AWS EC2 Dashboard. On the left, there's a navigation sidebar with various options like EC2 Dashboard, Instances, Images, Elastic Block Store, Network & Security, and Load Balancing. A large arrow points from the 'Instances' link in the sidebar to the 'Instances (running)' section in the main content area. The main content area has a title 'Resources' and a sub-section 'Instances (running)'. It displays the following data:

Category	Value
Instances	1
Auto Scaling Groups	0
Dedicated Hosts	0
Elastic IPs	0
Instances	1
Key pairs	2
Load balancers	0
Placement groups	0
Security groups	7
Snapshots	0
Volumes	1

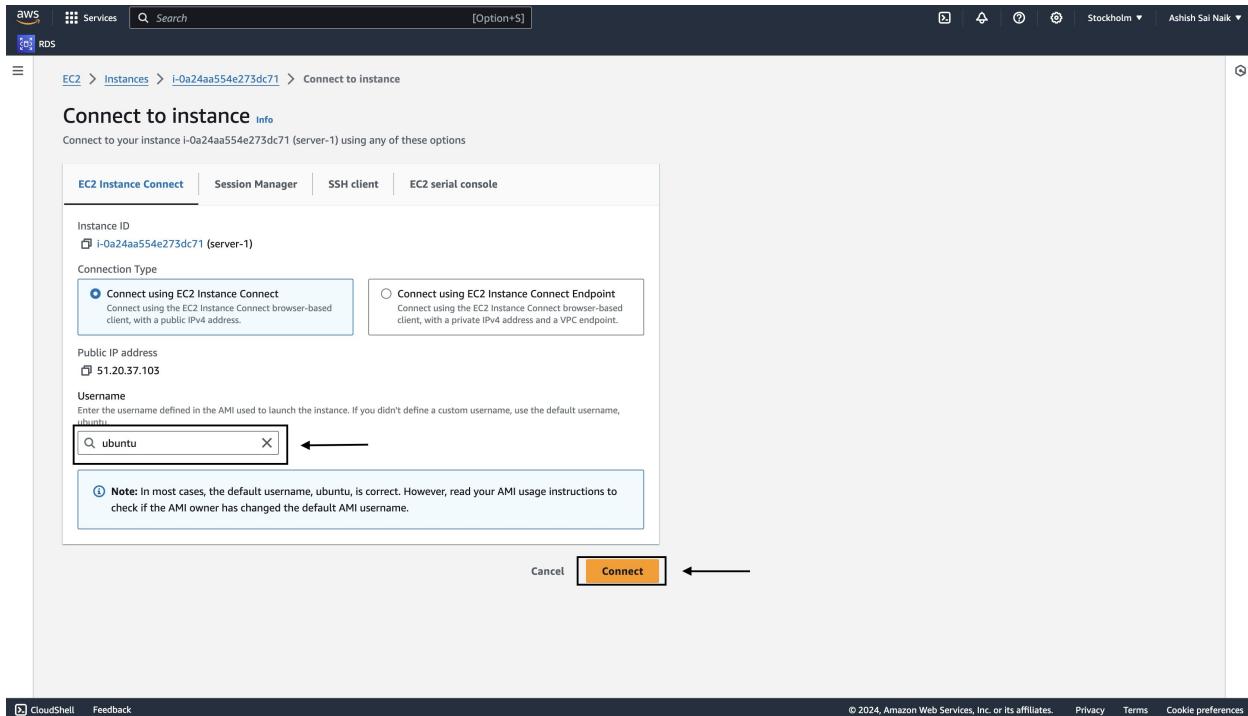
Below this, there are sections for 'Launch instance', 'Service health', 'Instance alarms', and 'Scheduled events'. To the right, there's a sidebar with 'EC2 Free Tier' information, 'Offer usage (monthly)', and 'Account attributes'.

After redirecting into the Instance(running), Select the Instance which we want to connect to our Django project, Then Click the "connect" .

The screenshot shows the AWS EC2 Instances page. On the left, a sidebar menu includes 'EC2 Dashboard', 'EC2 Global View', 'Events', 'Instances' (which is expanded), 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations', 'Images' (with 'AMIs' and 'AMI Catalog'), 'Elastic Block Store' (with 'Volumes' and 'Snapshots'), 'Lifecycle Manager', 'Network & Security' (with 'Security Groups', 'Elastic IPs', 'Placement Groups', 'Key Pairs', and 'Network Interfaces'), and 'Load Balancing'. The main content area displays a table titled 'Instances (1/1) Info' with one row for 'server-1'. The table columns include 'Name' (server-1), 'Instance ID' (i-0a24aa554e273dc71), 'Instance state' (Running), 'Instance type' (t3.micro), 'Status check' (2/2 checks passed), 'Alarm status' (View alarms), 'Availability Zone' (eu-north-1b), 'Public IPv4 DNS' (ec2-51-20-37-103.eu-n...), and 'Public IPv4 address' (51.20.37.103). A 'Connect' button is available in the top right of the table header. Below the table, a detailed view for 'i-0a24aa554e273dc71 (server-1)' is shown with tabs for 'Details', 'Status and alarms', 'Monitoring', 'Security', 'Networking', 'Storage', and 'Tags'. The 'Details' tab is selected, displaying fields like 'Instance summary' (with 'Public IPv4 address' 51.20.37.103), 'Instance state' (Running), 'Private IP DNS name (IPv4 only)' (ip-172-31-37-198.eu-north-1.compute.internal), 'Instance type' (t3.micro), and 'VPC ID'. Other tabs show 'AWS Compute Optimizer finding' and 'AWS Lambda function metrics'.

Once you clicked the connect , we will redirect to the below page .

Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu. then we are ready to connect.



After we connect, we will follow to the terminal

```

aws Services Search [Option+S] | RDS
EC2 Instances i-0a24aa554e273dc71 Connect to instance
Connect to instance Info
Connect to your instance i-0a24aa554e273dc71 (server-1) using any of these options
EC2 Instance Connect Session Manager SSH client EC2 serial console
Instance ID i-0a24aa554e273dc71 (server-1)
Connection Type
 Connect using EC2 Instance Connect
  Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.
 Connect using EC2 Instance Connect Endpoint
  Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.
Public IP address 51.20.37.103
Username
  Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.
  ubuntu
  ↪
  ⓘ Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.
Cancel Connect ↪

CloudShell Feedback | © 2024, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences
  
```

Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1008-aws x86_64)

* Documentation: <https://help.ubuntu.com>
 * Management: <https://landscape.canonical.com>
 * Support: <https://ubuntu.com/pro>

System information as of Wed Jun 5 12:17:59 UTC 2024

```

System load: 0.11 Temperature: -273.1 C
Usage of '/': 39.9% of 6.71GB Processor: 118
Memory usage: 35% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 172.31.37.198
  
```

* Ubuntu Pro delivers the most comprehensive open source security and compliance features.

<https://ubuntu.com/aws/pro>

Expanded Security Maintenance for Applications is not enabled.

26 updates can be applied immediately.
 To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
 See <https://ubuntu.com/esm> or run: sudo pro status

*** System restart required ***
Last login: Wed Jun 5 10:45:36 2024 from 13.48.4.202
ubuntu@ip-172-31-37-198:~\$

i-0a24aa554e273dc71 (server-1)
Public IPs: 51.20.37.103 Private IPs: 172.31.37.198

Install Packages

Since linux ami has its own python pre installed, I did not include it in my installations. I only added pip and django

```
sudo apt update  
sudo apt-get install python3-pip  
sudo pip3 install gunicorn  
sudo apt-get install supervisor  
sudo pip3 install django  
sudo yum install nginx
```

Once we are done with the installing the packages , then we are ready to Clone the Django project in the git terminal

```
git clone -b Branch-1 https://github.com/yaswanthkoneri/job-portal-django.git
```

Download the django

```
sudo apt install python3-pip -y  
pip install django
```

check, Does the git have been successful clone or not by

```
ls -lrt
```

this commada show the project name , alone with the total number of the files in it.

```
python3 manage.py makemigrations  
python3 manage.py migrate
```

This create the all the migrations file required to run the app

One last step and our Job-portal-django project will be live by creating the Admin user to run this app. On the terminal, type the following command and provide username, email, password and once again password

```
python3 manage.py createsuperuser
```

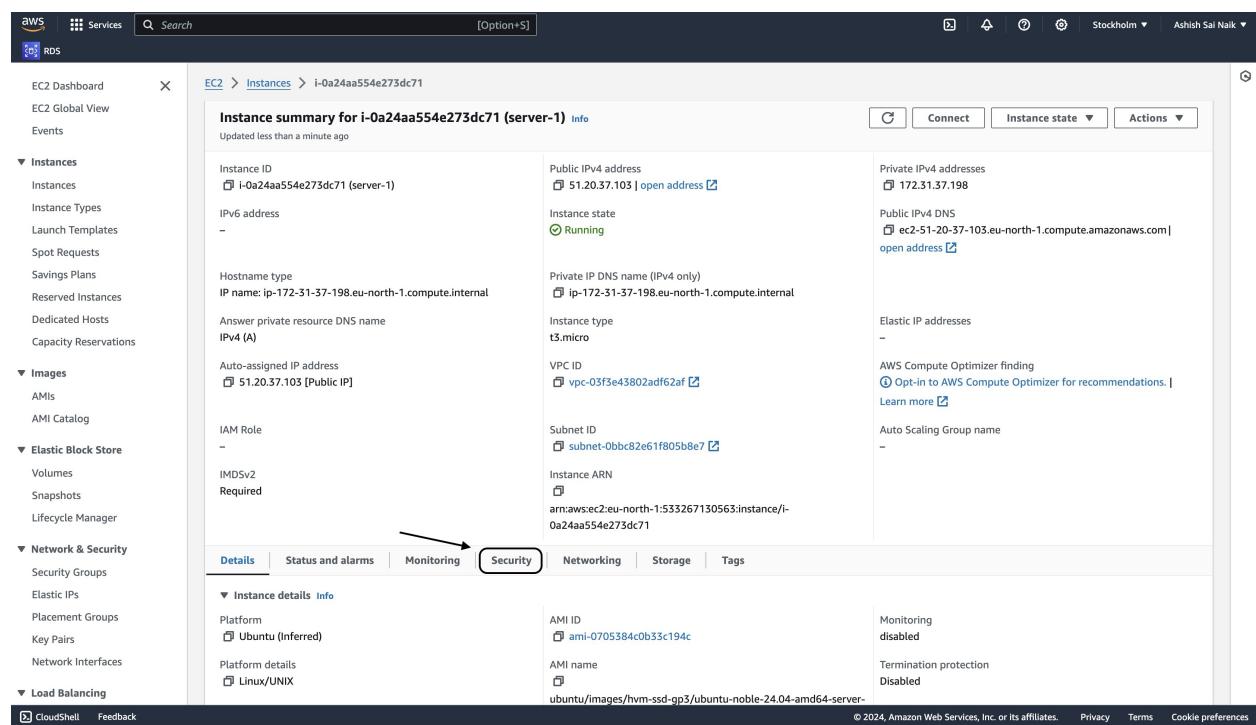
now its make app Live , we just need to start the server by the following the command line

```
python3 manage.py runserver
```

Once server hosted , we will head to the <http://127.0.0.1:8000/>

Since its still hosted in the local host so we have to change to the anywhere, we have to edit the security inbound rules, follow the step to change the inbound rule.

Step-1



The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various services like EC2 Dashboard, EC2 Global View, Events, Instances, Images, Elastic Block Store, Network & Security, and Load Balancing. The main content area shows an instance summary for 'i-0a24aa554e273dc71 (server-1)'. The 'Security' tab is highlighted with a black arrow pointing to it. The instance details include: Instance ID (i-0a24aa554e273dc71), Public IPv4 address (51.20.37.103), Instance state (Running), Private IP DNS name (ip-172-31-37-198.eu-north-1.compute.internal), Instance type (t3.micro), VPC ID (vpc-03f3e43802adfc62af), Subnet ID (subnet-0bbc82e61f805b8e7), Instance ARN (arn:aws:ec2:eu-north-1:533267130563:instance/i-0a24aa554e273dc71), IAM Role (None), IMDSv2 Required, and several other fields like Platform (Ubuntu (Inferred)), AMI ID (ami-0705384c0b33c194c), and AMI name (ubuntu/images/hvm-ssd-gp3/ubuntu-noble-24.04-amd64-server).

step-2

The screenshot shows the AWS EC2 Instances page with the Security tab selected. A specific instance is highlighted, showing its security group assignment. The 'Security groups' section lists 'sg-0ccdf4faa6e8d0dc (launch-wizard-2)'. Below it, the 'Inbound rules' and 'Outbound rules' sections show the configured security group rules.

Name	Security group rule ID	Port range	Protocol	Source	Security groups
-	sgr-0b778b514752767e9	8000	TCP	0.0.0.0/0	launch-wizard-2
-	sgr-0dde7cdcb502d2c40	22	TCP	0.0.0.0/0	launch-wizard-2

Name	Security group rule ID	Port range	Protocol	Destination	Security groups
-	sgr-09cf7ce01fbde23c4	All	All	0.0.0.0/0	launch-wizard-2

step-3

The screenshot shows the AWS Security Groups page for the group 'sg-0ccdf4faa6e8d0dc - launch-wizard-2'. The 'Inbound rules' tab is selected, displaying two entries. An arrow points to the 'Edit inbound rules' button.

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-0b778b514752767e9	IPv4	Custom TCP	TCP	8000	0.0.0.0/0
-	sgr-0dde7cdcb502d2c40	IPv4	SSH	TCP	22	0.0.0.0/0

step-4

Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0b778b514752767e9	Custom TCP	TCP	8000	Custom	0.0.0.0/0 X
sgr-0dde7cdcb502d2c40	SSH	TCP	22	Custom	0.0.0.0/0 X

Add rule

⚠️ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

Click on the Add rule ,Make sure in the Type select the "Custom TCP" and post range as 8000 and Then good to go with save rules.

After completing with changing inbound rule , we have to update the code in the code

from jobportal>>setting.py update the Allowed_Host to all the IP address,

```
ALLOWED_HOSTS = ['0.0.0.0', 'localhost',
                 '127.0.0.1', '51.20.37.103']
```

Then return to the terminal run the server by the following command line

```
python3 manage.py runserver 0.0.0.0:8000
```

Finally our project get live in the AWS Cloud

Page not found (404)

Directory indexes are not allowed here.

Request Method: GET
Request URL: http://51.20.37.103:8000/
Raised by: django.views.static.serve

Using the URLconf defined in jobportal.urls, Django tried these URL patterns, in this order:

1. admin/
2. jobs/
3. '^(<path>.*\$'

The empty path matched the last one.

You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.