A Mini Project Report

on

"HOSTING DJANGO WEBSITE USING AMAZON EC2"

Submitted to

CLOUD COMPUTING LAB
(20BT61231)

BACHELOR OF TECHNOLOGY

IN
INFORMATION TECHNOLOGY

Submitted by

P.Madhulika	**21121A1282**
P.Lakshmi Prasanna	**21121A1283**
P.Neelima	**21121A1284**
P.Madhusudhan	**21121A1285**
P.Bharath	**21121A1286**
P.Sri Sai Priya	**21121A1287**

Department of Information Technology
SREE VIDYANIKETHAN ENGINEERING COLLEGE
(AUTONOMOUS)
(Affiliated to JNTUA, Ananthapuramu, Approved by AICTE, Accredited by NBA & NAAC) Sree Sainath Nagar, Tirupati – 517 102, A.P., INDIA

2023-2024

ABSTRACT

This explores the process of deploying a Django web application on Amazon Elastic Compute Cloud (EC2), a popular cloud computing platform. Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Amazon EC2 provides scalable computing capacity in the cloud, allowing developers to deploy applications quickly and efficiently. The begins by outlining the prerequisites for deploying a Django application on Amazon EC2, including setting up an EC2 instance, configuring security groups, and installing necessary software packages. Next, it walks through the process of preparing the Django application for deployment, including configuring settings, managing static files, and setting up a database. Once the application is ready for deployment, the discusses various methods for deploying the Django application on Amazon EC2, including manual deployment via SSH and automated deployment using tools like AWS Elastic Beanstalk or AWS CodeDeploy. It covers best practices for managing deployment configurations, monitoring application performance, and ensuring security. Throughout the, real-world examples and best practices are provided to help developers understand the deployment process and overcome common challenges. By the end of the, readers will have a comprehensive understanding of how to deploy a Django web application on Amazon EC2, enabling them to leverage the scalability and flexibility of the cloud to host their Django projects efficiently.

INTRODUCTION

The integration of cloud computing has become a fundamental paradigm shift. Cloud platforms offer developers an unparalleled level of flexibility, scalability, and accessibility, enabling them to deploy and manage applications with unprecedented ease. Among these platforms, Amazon Web Services (AWS) has emerged as a leader, providing a comprehensive suite of cloud services that cater to the diverse needs of developers and businesses alike.

The landscape of web development has been transformed by the advent of cloud computing, offering developers new avenues for deploying and scaling applications.Among the prominent cloud platforms, Amazon Web Services (AWS) has emerged as a leader, providing a comprehensive suite of services to support developers in their endeavors. At the forefront of AWS services is Amazon Elastic Compute Cloud (EC2), which offers resizable compute capacity in the cloud, enabling developers to dynamically adjust their infrastructure to meet evolving demands.

At the heart of AWS's offerings lies Amazon Elastic Compute Cloud (EC2), a cornerstone service that revolutionizes the way compute resources are provisioned and managed in the cloud. EC2 offers developers resizable virtual servers, known as instances, which can be quickly scaled up or down to accommodate fluctuating workloads. This elasticity not only enhances operational efficiency but also empowers developers to optimize resource allocation and minimize costs.

The convergence of Django and AWS EC2 represents a symbiotic relationship that harnesses the strengths of both platforms to create a robust and scalable infrastructure for deploying web applications. By leveraging Django's simplicity and flexibility alongside EC2's scalability and reliability, developers can build and deploy sophisticated web applications that are capable of meeting the demands of modern users.

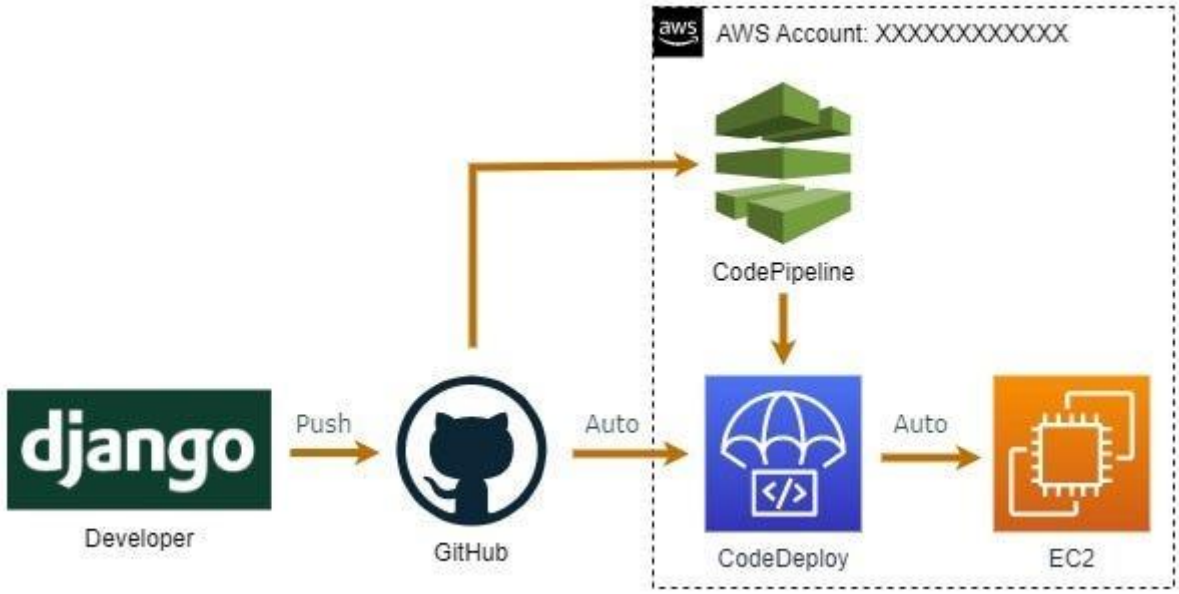
Concurrently, Django has risen to prominence as a popular web framework for Python developers, celebrated for its simplicity, flexibility, and scalability. Django empowers developers to rapidly build web applications while adhering to best practices in software design.

By combining the capabilities of Django with the scalability offered by AWS EC2,developers gain access to a powerful platform for deploying robust web applications on a flexible and reliable infrastructure. By leveraging the strengths of both Django and AWS EC2, developers can create and deploy web applications that are not only feature-rich but also capable of handling the demands of modern web traffic.

- Setting up an Amazon EC2 instance and configuring security groups.
- Preparing a Django application for deployment, including managing static files and configuring the database.
- Exploring various deployment methods, ranging from manual SSH deployment to automated deployment using AWS services like Elastic Beanstalk or CodeDeploy.
- Best practices for managing deployment configurations, monitoring application performance, and ensuring security.
- Real-world examples and practical tips to address common challenges encountered during deployment.

The developers will have gained a thorough understanding of deploying a Django web application on Amazon EC2. Armed with this knowledge, they will be well-equipped to leverage the scalability and flexibility of the cloud to efficiently host their Django projects.

RESOURCE DIAGRAM



RESOURCES LIST

- EC2
- Django
- VPC

PROCEDURE

Step 1: Create AWS account

Visit AWS Console to create your account and log in.

Step 2: Create a virtual Private Network(VPC)

- Name tag:Give vpc name
- IPV4 CIDR block :Define IP address range for :name,eg:”10.0.0.0/16”
- Click on create vpc & after launch-VPC-ID will be generated

Step 3: Create Two Subnets and Internet Gateway

- VPC: Select the VPC which created earlier (JNETWORK)
- IGW- Go to VPC dashboard.,Click the "Create Internet Gateway" button
- click "Attach to VPC." Choose VPC & click "Attach.".

Step 4: Create a Route Table & Associate Private Subnets 1 & Subnet2

- In the VPC dashboard, click on "Route Tables"
- Give it a name, e.g., "RT-PublicRouteTable," and select VPC (JNETWORK) & Create.
- Select the route table just created, click the "Routes" tab, and then click "Edit routes."
- Add a route with destination "0.0.0.0/0" and target as the Internet Gateway (MyInternetGateway-IGW) created earlier & Save the changes.
- Click on Subnet Associations & Add “ Subnet 1 & Subnet2” which created earlier & Save the changes.

Step 5: EC2 instance Setup

- Go to the EC2(SUBNET1 & SUBNET2) dashboard & Click on "Launch Instance" to create a new EC2 instance.
- Choose an Amazon Machine Image (AMI) for instance, e.g., “ Amazon Linux 2”.
- Select an instance type and configure instance details and Configure security groups to allow HTTP (port 80) , SSH (port 22) & HTTPS (port 443) traffic.
- Review and launch the instance. existing key pair(password.ppk) for SSH access.Launch the instance.

- Connect to EC2 (SUBNET 1) & EC2 (SUBNET2) instance,SSH into EC2 instance using the key pair (password.ppk)
    - EC2(SUBNET 1) - Connect is Successful
    - EC2(SUBNET 2) - Connect is Successful

Step 6: Deploy Django Project on AWS EC2 instance

Run these commands in command prompt after connecting EC2

- sudo apt update
- git clone https://github.com/yeshwanthlm/django-on-ec2.git
- -lrt
- cd django-on-ec2
- sudo apt install python3-pip –y
- pip install django
- python3 manage.py migrate
- python3 manage.py createsuperuser
    - Enter email and password (superuser created suceesfully)
- python3 manage.py runserver

Please copy the following link (http://127.0.0.1:8000/) and paste it into your browser's address bar. After browsing to it, the to-do list(output) will open.

SOURCE CODE

```
<title>Todo list</title>
<div class="container">
  <div class="row">
  <div class="offset-md-2 col-lg-9">
   <div class="page-header">
    <h1>
     Todo List
    </h1>
   </div>
  </div>
 </div>
 <div class="row">
  <div class="offset-md-2 col-lg-9">
   <form method="post" action="{% url 'todos:add' %}">
    <div class="form-row">
     <div class="col-md-6">
      <input  type="text"  class="form-control"  name="title"  placeholder="Do
laundry" required>
     </div>
     <div class="col-md-6">
      <button type="submit" name="submit" class="btn btn-outline-primary">
       Add
      </button>
     </div>
    </div>
   </form>
  </div>
 </div>
 <hr />
 <div class="row">
  <div class="offset-md-2 col-lg-6">
   <div class="list-group">
    {% for todo in todo_list %}
```

```
        <div class="list-group-item {% if todo.isCompleted %} todo-complete {% endif
%}">
          <form style="display: inline;" method="post" action="{% url 'todos:update'
todo.id %}">

            <input type="checkbox" name="isCompleted"
onchange="this.form.submit()" {% if todo.isCompleted %} checked
              {% endif %} class="todo-status-checkbox"
              title="{% if not todo.isCompleted %} mark as done {% else %} mark undone
{% endif %}">
          </form>
          {{ todo.title }}
          <a href="{% url 'todos:delete' todo.id %}" title="Delete">
            <i class="far fa-trash-alt"></i>
          </a>
        </div>
      </div>
    </div>
  </div>
</div>
```

CONCLUSION

In conclusion, hosting a Django website on Amazon EC2 offers a robust and scalable solution for deploying web applications. Throughout this process, we have explored the steps required to set up an EC2 instance, configure it to run Django, and deploy our website effectively. By leveraging the flexibility and scalability of EC2, we can easily adjust resources to accommodate varying levels of traffic and ensure optimal performance for our Django application. Furthermore, utilizing Amazon EC2 provides access to a wide range of AWS services that can enhance the functionality and reliability of our Django website. Services such as Amazon RDS for database management, Amazon S3 for static file storage, and Amazon CloudFront for content delivery can be seamlessly integrated to improve the overall performance and user experience. Overall, hosting a Django website on Amazon EC2 empowers developers to create robust, scalable, and reliable web applications while benefiting from the flexibility and scalability of the AWS cloud infrastructure. With proper configuration and optimization, Amazon EC2 serves as an excellent platform for deploying Django websites without compromising on performance or reliability.

REFERENCES

- Amazon Web Services Documentation
- Django Deployment on EC2
- Automating Django Deployment with Ansible
- Tutorial: Deploying Django on AWS
- ProgramInk-Deploying Django App