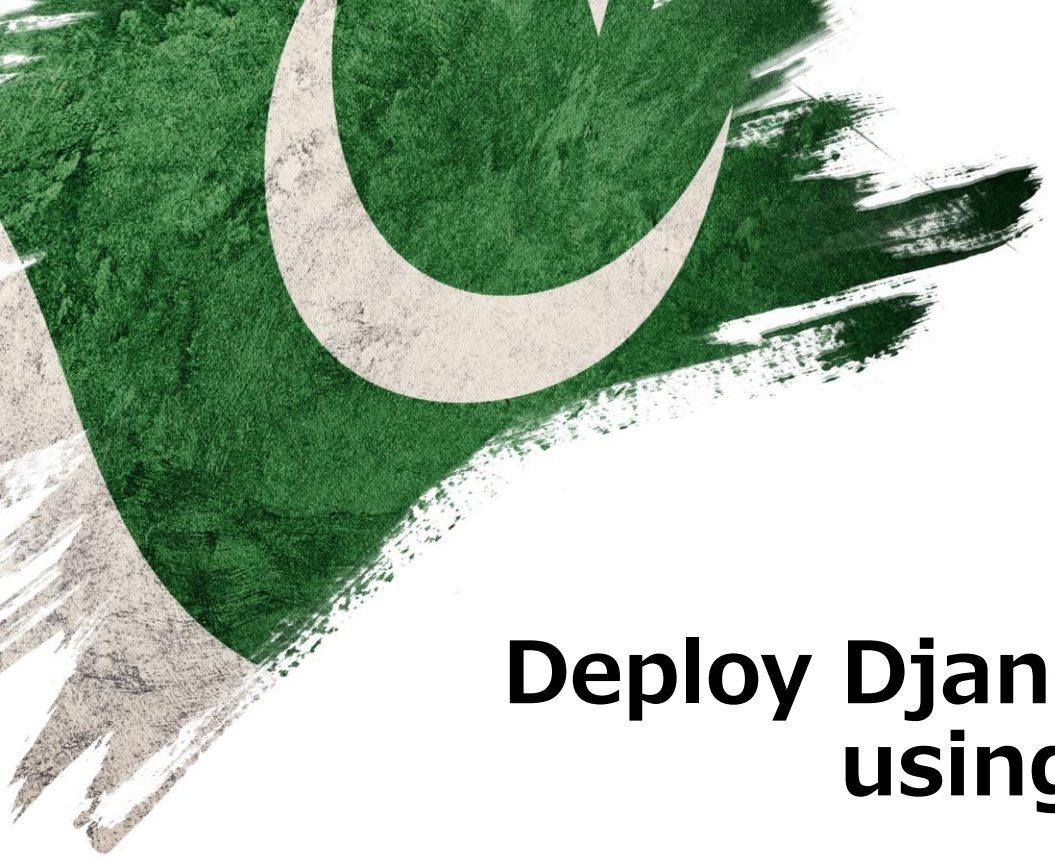

Deploy Django application on AWS using Bash scripting

Class 23

2/8/2025

Acknowledgement

The series of the IT & Japanese language course is Supported by AOTS and OEC.

Ministry of Economy, Trade and Industry

Overseas Employment Corporation

What you have Learnt Last Week

We were focused on following points.

- Usage of control and loop flow statement
- Performing Linear Algebra in Numpy
- Why Requirement Analysis is so important in the process?
- Software development Life cycle
- Importance of Security compliance
- Introduction of Bash Scripting
- Introduction of Ansible, docker and docker compose
- API testing with Postman

What you will Learn Today

We will focus on following points.

1. Creating an EC2 instance, configuring security groups, and preparing the instance for Django deployment
2. Create a bash file by using vi editor
3. Write a bash scripts to automate the installation of Python, pip, virtual environments, and required Django dependencies on the server
4. Deploying and Running the Django Application with Bash
5. Quiz
6. Q&A Session

Launching an EC2 Instance

Choose the Right Configuration

- Go to **AWS Management Console → EC2 → Launch Instance**
- **Select AMI:** Ubuntu 20.04 (Free Tier eligible)
- **Instance Type:** t2.micro (1 vCPU, 1GB RAM)
- **Key Pair:** Create or use existing .pem file for SSH access
- **Elastic IP:** Allocate to keep a permanent public IP

Configuring Security Groups

Control Access to Your Instance

Security Group = Virtual Firewall

Add rules to allow specific traffic:

SSH (22): Restrict to your IP for secure access

HTTP (80): For web traffic

HTTPS (443): For secure web apps

Custom Port (e.g., 8000): For app testing

Connecting via SSH

Accessing Your EC2 Securely

Download .pem key if new key pair created

Set permissions:

chmod 400 my-key.pem

Connect via terminal:

ssh -i my-key.pem ubuntu@<public-ip>

You are now inside the EC2 instance!

Why Restrict SSH to Your IP?

Security Best Practice

- Prevents unauthorized access
- Limits SSH to your machine’s IP
- Reduces risk of brute-force attacks
- Always update the rule if your IP changes

Common Mistakes to Avoid

Troubleshooting Tips

- **Permission denied (publickey):** Check .pem file permissions
- **Connection timed out:** Ensure SSH port (22) is open
- **Wrong username:** For Ubuntu use ubuntu@ip (Amazon Linux uses ec2-user@ip)
- **Forgot Elastic IP:** Public IP changes on reboot without Elastic IP

Update Your EC2 System

Keep Your Server Secure & Up-to-date

After connecting to EC2 via SSH, update packages:

sudo apt update && sudo apt upgrade -y

Why?

Fixes security vulnerabilities

Updates package versions

Ensures a stable environment

Install Python3 and Pip

Core Requirements for Django

Install Python & pip:

sudo apt install python3 python3-pip -y

Verify installation:

python3 --version

pip3 --version

Set Up Virtual Environment

Clone Your Project from GitHub

Install virtualenv:

sudo pip3 install virtualenv

Create and activate:

virtualenv venv

source venv/bin/activate

Benefits:

Keeps project dependencies isolated

Avoids version conflicts

Install Git

Clone Your Project from GitHub

Install Git:

sudo apt install git -y

Verify:

git --version

Install Nginx and Gunicorn

Serve Django App in Production

Install:

sudo apt install nginx -y
sudo pip3 install gunicorn

Nginx = Web server

Gunicorn = WSGI server to run Django

Configure UFW Firewall

Allow Necessary Ports

Install and enable UFW:

sudo apt install ufw -y

sudo ufw allow OpenSSH

sudo ufw allow 'Nginx Full'

sudo ufw enable

Ensures only required traffic is allowed (SSH, HTTP, HTTPS)

Install PostgreSQL

For Production Database

Install PostgreSQL:

sudo apt install postgresql postgresql-contrib libpq-dev -y

Install driver in virtual environment:

pip install psycopg2-binary

Use for better performance and scalability in Django projects

Why Use a Deployment Script?

Automate Django Setup & Handle Errors

- Automates repetitive deployment steps.
- Helps maintain consistency across environments.
- Allows adding error handling for failed commands.

Used to:

Update & Install
Clone or Update Project
Virtual Environment
Django Commands
Restart Services

Create a Bash File Using vi

Start Your Deployment Script

Step1: SSH into your EC2 Instance
ssh -i key.pem ubuntu@<EC2-IP>

Step2: Create a new bash file
vi deploy.sh

Step3: Inside vi press i to insert and add:
#!/bin/bash echo "Starting Deployment Script"

Step4: Save and exit vi:
Press Esc → :wq → Enter

Step5: Make it executable:
chmod +x deploy.sh

Installing Python, Pip & Virtualenv

Automate Django Environment Setup

Add to deploy.sh:

```
echo "Updating packages"
sudo apt update && sudo apt upgrade -y || exit 1

echo "Installing Python & Pip"
sudo apt install python3 python3-pip -y || exit 1

echo "Installing Virtualenv"
pip3 install virtualenv || exit 1
```

Why?

- Python3 for Django
- Pip for package management
- Virtualenv keeps dependencies isolated

Setting Up Virtual Environment & Requirements

Prepare Your Project Environment

Inside deploy.sh:

```
if [ ! -d "venv" ]; then
  echo "Creating virtual environment"
  virtualenv venv || exit 1
fi

echo "Activating environment"
source venv/bin/activate
```

Install dependencies from requirements.txt:

```
echo "Installing project dependencies"
pip install -r requirements.txt || exit 1
```

Cloning the Django Notes App

Get Code from GitHub Repo

Add to deploy.sh:

```
if [ ! -d "django-notes-app" ]; then
  echo "Cloning Django Notes App"
  git clone https://github.com/LondheShubham153/django-notes-app || exit 1
else
  echo "Updating existing repo"
  cd django-notes-app && git pull || exit 1
fi
```

Checks if folder exists → clones or updates repo.

Running Migrations

Set Up Database Tables

Inside deploy.sh:

```
cd django-notes-app


echo "Running migrations"
python manage.py makemigrations || exit 1
python manage.py migrate || exit 1
```

Ensures database schema is up-to-date.

Collecting Static Files

Prepare CSS/JS for Production

```
echo "Collecting static files"

python manage.py collectstatic --noinput || exit 1
```

Gathers all static files in one folder for serving via Nginx.

Restarting Gunicorn & Nginx

Make Changes Live

```
echo "Restarting Gunicorn & Nginx"

sudo systemctl restart gunicorn || exit 1

sudo systemctl restart nginx || exit 1
```

Reloads app server and web server with new code.

Putting It All Together

Complete deploy.sh Script

```
#!/bin/bash
echo "Deploying Django Notes App"

# Update & Install
sudo apt update && sudo apt upgrade -y
sudo apt install python3 python3-pip -y
pip3 install virtualenv

# Clone or Update Project
if [ ! -d "django-notes-app" ]; then
  git clone https://github.com/LondheShubham153/django-notes-app

else
  cd django-notes-app && git pull
fi
```

Putting It All Together

Complete deploy.sh Script

```
# Virtual Environment
cd django-notes-app
virtualenv venv
source venv/bin/activate
pip install -r requirements.txt

# Django Commands
python manage.py makemigrations
python manage.py migrate
python manage.py collectstatic --noinput

# Restart Services
sudo systemctl restart gunicorn
sudo systemctl restart nginx
echo "✅ Deployment Completed"
```

Running the Script

Deploy in One Command

Run script:
./deploy.sh

Check app in browser using EC2 Public IP:
http://<EC2-IP>:8000

Verify services:
sudo systemctl status gunicorn sudo systemctl status nginx

Key Takeaways

Automating Django Deployment

- vi editor for creating bash scripts.
- Automate environment setup + migrations + static files.
- Restart services for changes to take effect.
- One command = full deployment.

Assignment

Assignment

1. Create a bash script and deploy any Django application

Quiz Section

Quiz

Everyone student should click on submit button before time ends otherwise MCQs will not be submitted

[Guidelines of MCQs]

1. There are 20 MCQs
2. Time duration will be 10 minutes
3. This link will be share on 12:25pm (Pakistan time)
4. MCQs will start from 12:30pm (Pakistan time)
5. This is exact time and this will not change
6. Everyone student should click on submit button otherwise MCQs will not be submitted after time will finish
7. Every student should submit Github profile and LinkedIn post link for every class. It include in your performance

Assignment

Assignment should be submit before the next class

[Assignments Requirements]

1. Create a post of today's lecture and post on LinkedIn.
2. Make sure to tag @Plus W @Pak-Japan Centre and instructors LinkedIn profile
3. Upload your code of assignment and lecture on GitHub and share your GitHub profile in respective your region group WhatsApp group
4. If you have any query regarding assignment, please share on your region WhatsApp group.
5. Students who already done assignment, please support other students

Q&A Session

ありがとうございます。
Thank you.
شکریا

For the World with Diverse Individualities