DEVOPS PROJECT REPORT

Project Title: Online Job Portal

Section: S-11(Cluster-1)

Student Register numbers: 2200031007 – Annepu Uday Kumar
2200031949 – Vuyyuru Daiva Lokesh
2200032761 – Suravarapu Deepthi

Instructor Name: G.Bindu

Course Coordinator: G.Bindu

Course Title: Cloud DevOps

Course Code: 22SDC105A

Table of Contents

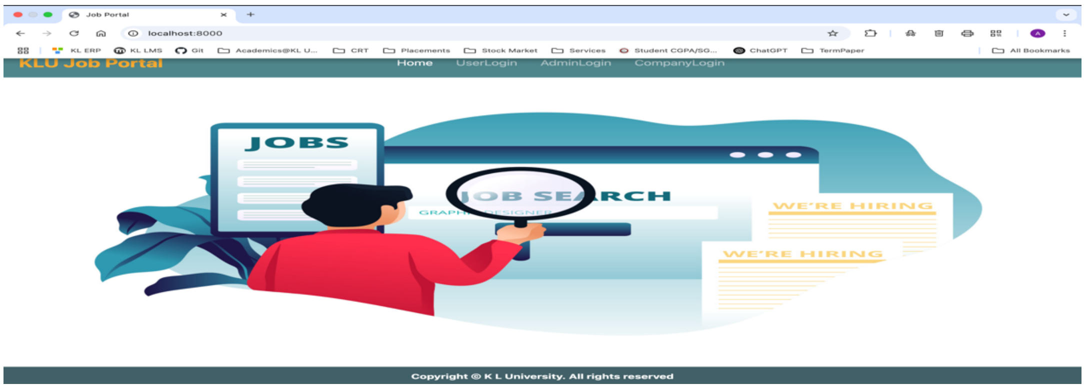
1. Introduction

The **Online Job Portal** is a web-based platform designed to facilitate job opportunities for students and graduates by connecting them with recruiters and employers. This portal serves as a bridge between students seeking internships, part-time jobs, or full-time employment and companies looking for skilled candidates from the university.

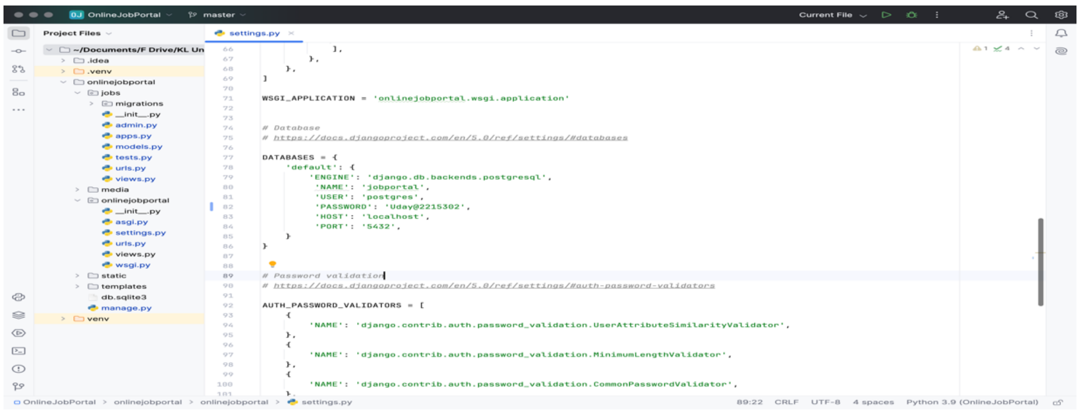
2. Project Overview

The system is built using a **modern technology stack** to ensure seamless operation and user-friendly functionality:

- ✓ **Frontend:** Developed using **HTML, CSS, and JavaScript**, providing a responsive and interactive interface for students, recruiters, and University administrators.

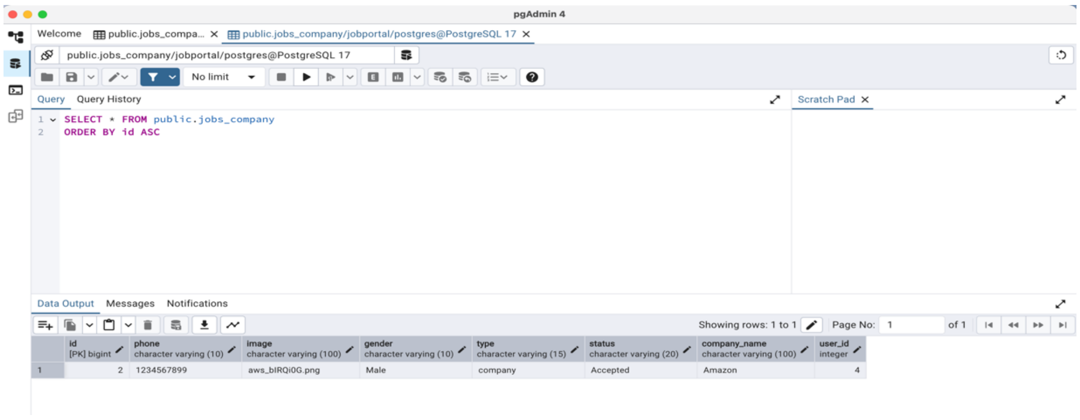


- ✓ **Backend:** Implemented using **Django**, a powerful Python web framework known for its scalability, security, and ease of development.



```
                ],
            },
        },
]

WSGI_APPLICATION = 'onlinejobportal.wsgi.application'


# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
         'NAME': 'jobportal',
         'USER': 'postgres',
         'PASSWORD': 'Uday@2215302',
         'HOST': 'localhost',
         'PORT': '5432',
    }
}


# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
```

- ✓ **Database:** Utilizes **PostgreSQL**, a robust relational database system, to efficiently manage job listings, student profiles, applications, and employer data.



```
SELECT * FROM public.jobs_company
ORDER BY id ASC
```

	id [PK] bigint	phone character varying (10)	image character varying (100)	gender character varying (10)	type character varying (15)	status character varying (20)	company_name character varying (100)	user_id integer
1	2	1234567899	aws_bIRQi0G.png	Male	company	Accepted	Amazon	4

3. Technologies and Versions

- **Frontend Technologies:**
  - ✓ HTML5
- **Backend Technologies:**
  - ✓ Django
  - ✓ Version: v4.2
- **Database Technologies:**
  - ✓ PostgreSQL
  - ✓ Version: v17

4. Business Logic Explanation

- Describe the core logic and features of the project.
- How the system behaves and handles requests.
- Business rules implemented in the backend.
- Describe any algorithms or processes involved in handling user data or inputs.

5. Tools Used in DevOps Pipeline

- **GitHub:**
  - ✓ Git is a **distributed version control system** that tracks changes in source code during software development. It's essential in **DevOps pipelines** for managing code history, collaboration, and triggering automated workflows.
    - **Source Code Management**: Git stores the entire history of code changes.
    - **Triggering CI/CD Pipelines**: Tools like **Jenkins**, **GitHub Actions**, or **GitLab CI** watch Git repositories for events like push, pull request, or merge. These events automatically trigger builds, tests, or deployments.
    - **Rollback Capabilities**: Developers can revert to previous versions if a deployment fails.
    - **Branching Strategies**: Git enables teams to use feature branches, hotfix branches, or GitFlow for structured collaboration.

- **Jenkins:**
  - ✓ Jenkins is an open-source automation server used for building, testing, and deploying code continuously in a DevOps pipeline.
    - **Automation of Builds:** Automatically compiles, tests, and packages the application when changes are pushed.
    - **CI/CD Integration:** Integrates with tools like Git, Docker, Kubernetes, and Ansible to support end-to-end delivery.
    - **Pipeline as Code:** Supports scripted and declarative pipelines using Jenkins file for consistent and version-controlled workflows.
    - **Plugin Support:** Offers 1800+ plugins to integrate with almost any DevOps tool.
    - **Monitoring & Notifications:** Sends build status via email, Slack, or dashboards for team visibility.

- **Bash Scripting:**

  ✓ Bash scripting is a powerful way to automate tasks on Unix-based systems (like Linux and macOS). A Bash script is simply a text file containing a series of commands that are executed in sequence by the Bash shell.

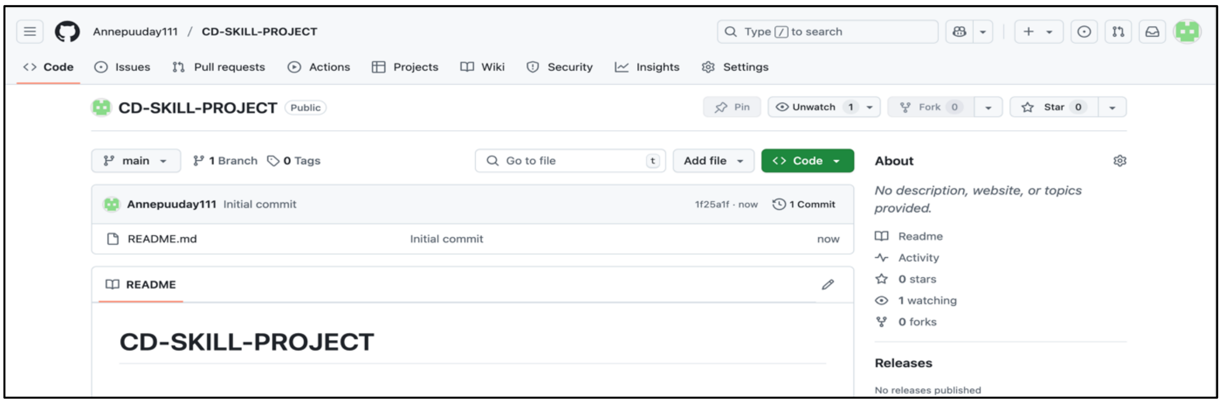
  Automation with Bash scripts allows users to:

  - Save time by executing repetitive tasks with a single command.
  - Reduce manual errors.
  - Ensure consistent execution of procedures.
  - Chain multiple commands and workflows into a single, manageable unit.

  Bash scripts are commonly used in **DevOps**, **System Administration**, and **Software Development** pipelines to control and automate workflows.

6. Execution of Tools

- **Terraform:**



✓ Create a folder for Terraform project

✓ Create a file named **main.tf** and Write the following inside **main.tf:**

```
provider "github" {
token = "your_github_token"
owner = "your_github_username"
}

resource "github_repository" "cd_skill_project" {
name        = "CD-SKILL-PROJECT"
description = "Repository for Continuous Delivery Skill Project"
visibility  = "public"
}
```

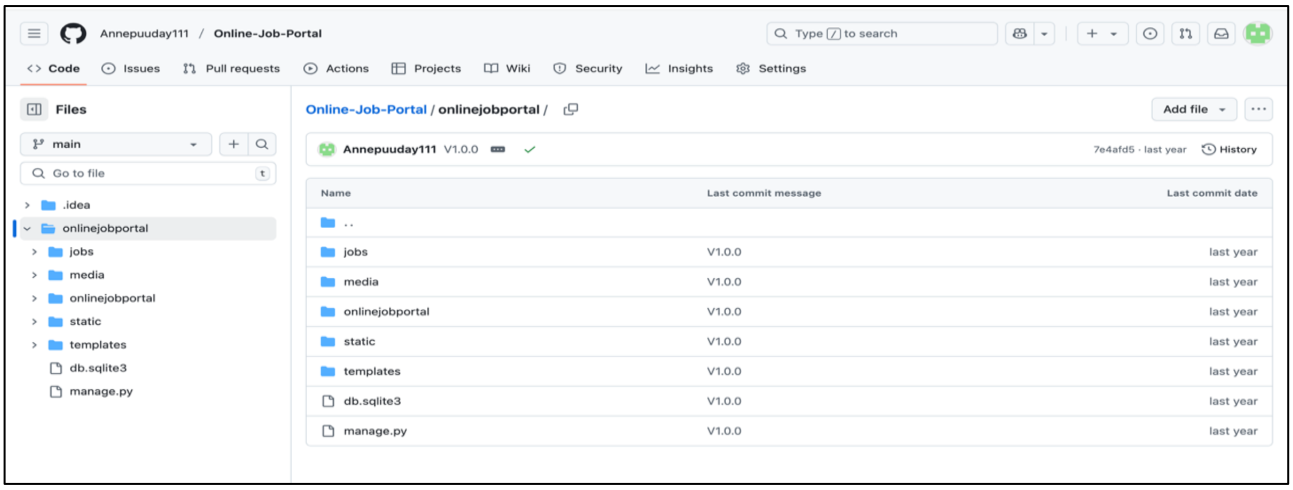
✓ Open terminal and navigate to the project folder

```
Run command: terraform init
Run command: terraform plan
Run command: terraform apply
```

✓ Confirm with "yes" when prompted

✓ Repository "CD-SKILL-PROJECT" will be created on GitHub

- **GitHub:**


Annepuuday111 / Online-Job-Portal

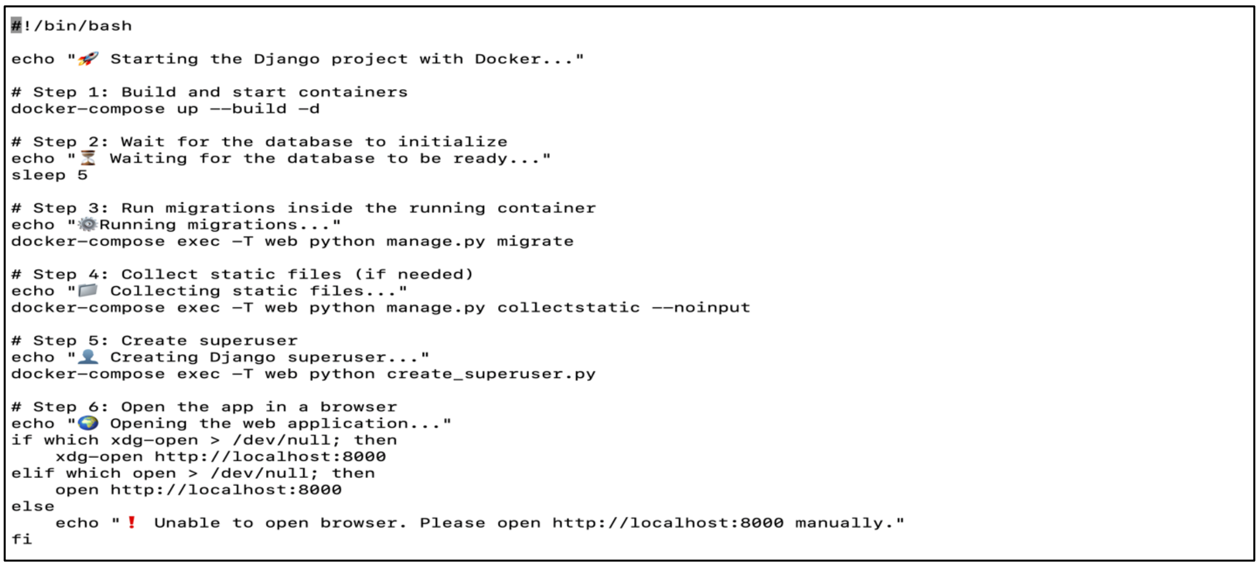
Online-Job-Portal / onlinejobportal /

Annepuuday111 V1.0.0 — 7e4afd5 · last year — History

Name	Last commit message	Last commit date
..		
jobs	V1.0.0	last year
media	V1.0.0	last year
onlinejobportal	V1.0.0	last year
static	V1.0.0	last year
templates	V1.0.0	last year
db.sqlite3	V1.0.0	last year
manage.py	V1.0.0	last year

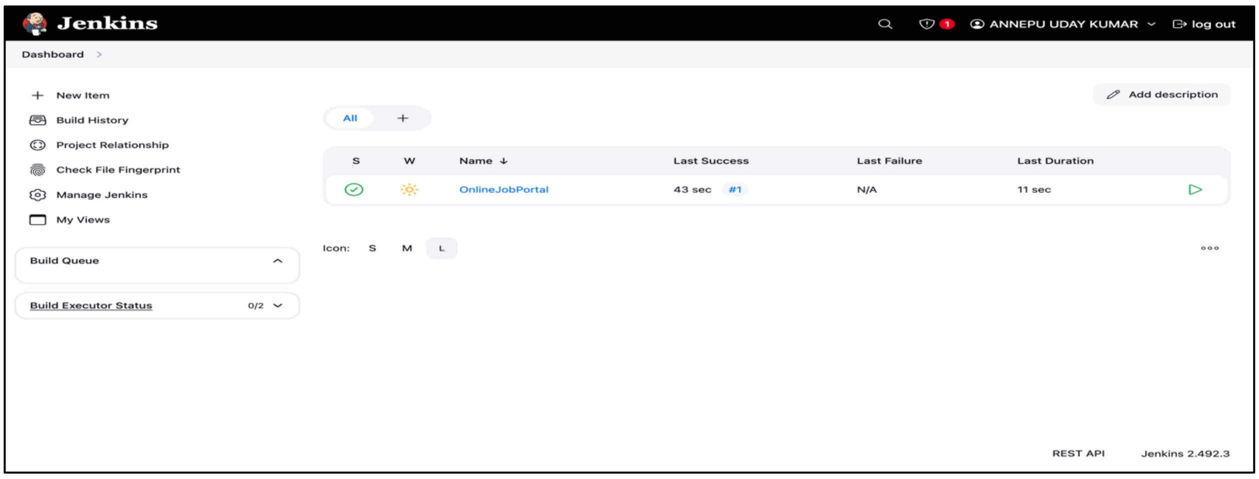
- ✓ git init
- ✓ git branch -m main
- ✓ git add .
- ✓ git commit -m "Version1"
- ✓ git remote add origin https://github.com/annepuuday/CD-SKILL-PROJECT.git
- ✓ error: remote origin already exists.
- ✓ git remote remove origin
- ✓ git remote add origin https://github.com/annepuuday/CD-SKILL-PROJECT.git
- ✓ git remote -v
- ✓ git push --set-upstream origin main

- **Bash Scripting:**


```
#!/bin/bash

echo "🚀 Starting the Django project with Docker..."

# Step 1: Build and start containers
docker-compose up --build -d

# Step 2: Wait for the database to initialize
echo "⏳ Waiting for the database to be ready..."
sleep 5

# Step 3: Run migrations inside the running container
echo "⚙️Running migrations..."
docker-compose exec -T web python manage.py migrate

# Step 4: Collect static files (if needed)
echo "📁 Collecting static files..."
docker-compose exec -T web python manage.py collectstatic --noinput

# Step 5: Create superuser
echo "👤 Creating Django superuser..."
docker-compose exec -T web python create_superuser.py

# Step 6: Open the app in a browser
echo "🌍 Opening the web application..."
if which xdg-open > /dev/null; then
    xdg-open http://localhost:8000
elif which open > /dev/null; then
    open http://localhost:8000
else
    echo "❗ Unable to open browser. Please open http://localhost:8000 manually."
fi
```

- ✓ docker-compose up --build -d
- ✓ docker-compose exec -T web python manage.py migrate
- ✓ docker-compose exec -T web python manage.py collectstatic --noinput
- ✓ docker-compose exec -T web python create_superuser.py

- **Jenkins:**



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	OnlineJobPortal	43 sec #1	N/A	11 sec

- ✓ Open Jenkins Dashboard
- ✓ Click "New Item"
- ✓ Enter job name and select "Freestyle project", then click OK
- ✓ Scroll to "Source Code Management"
- ✓ Select "Git"
- ✓ Enter Repository URL: https://github.com/your-username/your-repo.git
- ✓ Set Branch to build: main
- ✓ Scroll to "Build"
- ✓ Click "Add build step" → select "Execute shell" (or "Execute Windows batch command")
- ✓ Click "Save"
- ✓ Click "Build Now" to test your job
- ✓ Check "Console Output" under Build History for logs.

7. Survey on DevOps Tool Popularity

✓ To analyse the current landscape of DevOps tools, a survey was conducted using LinkedIn insights, Stack Overflow Developer Surveys, and reports from industry leaders like GitLab and the CNCF. The goal was to determine the popularity and adoption trends of various tools.

Key Findings:

- **Git** remains the most widely used version control system, with over **85%** of developers adopting it due to its distributed architecture and robust branching/merging capabilities.
- **Docker** is used by more than **75%** of DevOps teams for containerization, enabling consistent development environments and faster deployment cycles.
- **Kubernetes** has seen exponential growth, with **60%+** adoption, especially in cloud-native microservices orchestration.
- **Terraform** adoption is rising rapidly, reaching nearly **40%**, as organizations move toward Infrastructure-as-Code (IaC) practices.
- **Jenkins**, while facing competition from newer CI/CD tools, is still used by **50%+** of teams for continuous integration due to its extensibility and large plugin ecosystem.

8. Challenges and Solutions

Challenges Encountered:

- **Tool Integration Issues**: Conflicts between Kubernetes, Jenkins, and Docker during pipeline execution.
- **Network Configuration Errors**: Security group misconfigurations and port binding conflicts delayed deployment.
- **Complexity in IaC Setup**: Writing reusable and modular Terraform code proved difficult initially.

Solutions Implemented:

- Used Docker Compose for local testing to identify tool compatibility issues early.
- Standardized VPC and subnet configurations using Terraform modules to streamline network setups.
- Leveraged community modules and best practices to simplify IaC development.

Lessons Learned:

- Start small, test locally, and scale DevOps automation iteratively.
- Documentation and community forums are vital resources when troubleshooting.
- Observability (logs, metrics) is crucial to debugging CI/CD pipelines efficiently.

9. Conclusion

✓ This project successfully implemented a DevOps pipeline using tools like Git, Docker, Jenkins, Ansible, and Kubernetes. The integration of these tools significantly improved development velocity, deployment frequency, and system reliability.

Impact:

- Reduced manual deployment time by over 70%.
- Increased team collaboration via Git-based workflows.
- Achieved consistent infrastructure provisioning using IaC.

Future Improvements:

- Migrate to **GitHub Actions** or **GitLab CI** for more modern, cloud-native CI/CD experiences.
- Add **monitoring tools** (e.g., Prometheus, Grafana) for better observability.
- Introduce **security automation** in CI/CD for vulnerability scanning.

10. References

- Git Documentation – https://git-scm.com/doc
- Terraform Docs – https://www.terraform.io/docs
- Ansible Docs – https://docs.ansible.com
- Docker Documentation – https://docs.docker.com
- Kubernetes Docs – https://kubernetes.io/docs
- Jenkins User Documentation – https://www.jenkins.io/doc
- GitLab DevOps Survey 2024, 2025
- Stack Overflow Developer Survey 2024, 2025