Configuration Manual

MSc Research Project
Cloud Computing

Venkata Ratnam Atyam
Student ID: x23291788

School of Computing
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Venkata Ratnam Atyam
Student ID:	x23291788
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research project
Supervisor:	Shaguna Gupta
Submission Due Date:	15/09/2025
Project Title:	Automating UI Testing in CI/CD Pipelines Using Selenium for Cloud-Native Applications
Word Count:	1738
Page Count:	10

Signature:	Venkata Atyam
Date:	13th September 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Venkata Ratnam Atyam
x23291788

1 Introduction

This configuration manual provides step-by-step instructions to set up, run, and maintain a Django web application with Selenium UI tests integrated into a GitHub Actions CI/CD pipeline, deploying to an AWS EC2 instance behind Gunicorn + NGINX. It includes local setup, CI runner setup, server configuration, deployment automation, artifacts, and troubleshooting. **Scope:**

- Local development and test execution
- CI workflow with test artifacts (pytest HTML, JUnit XML, screenshots on failure)
- Secure deployment to EC2 using SSH
- Production stack: Gunicorn (WSGI), NGINX (reverse proxy), systemd service

2 System Configuration

2.1 Software Requirements

- Python: 3.10+
- Django: 4.x (or your project version)
- Selenium WebDriver + ChromeDriver/GeckoDriver
- PyTest (+ `pytest-html`)
- GitHub Actions
- AWS EC2 (Amazon Linux 2023)
- NGINX + Gunicorn
- Git, OpenSSH

2.2 Hardware/Hosting

- Local: laptop with $\geq$ 8GB RAM
- Server: `t3.small` (2 vCPU, 2GB RAM) or better; 20GB EBS; public/Elastic IP

3 Project Structure (expected)

```
.
|-- .github/
|   `-- workflows/
|       `-- ci.yml             # CI/CD workflow
|-- tests/                     # Selenium tests
|   `-- test_expense_tracker.py
|-- expense_tracker/           # Django project package
|-- manage.py
|-- requirements.txt
`-- README.md
```

Adjust paths if your repository differs.

4 Local Application Setup and Run

4.1 Clone and enter repository

```
git clone <https://github.com/venkat2356/selenium_cicd2.git>
cd <selenium_cicd2>
```

4.2 Create and activate virtual environment

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

4.3 Install dependencies

```
python -m pip install --upgrade pip
pip install -r requirements.txt
```

4.4 Run database migrations

```
python manage.py migrate
```

4.5 Run development server

```
python manage.py runserver
```

4.6 Run Selenium tests locally (headless)

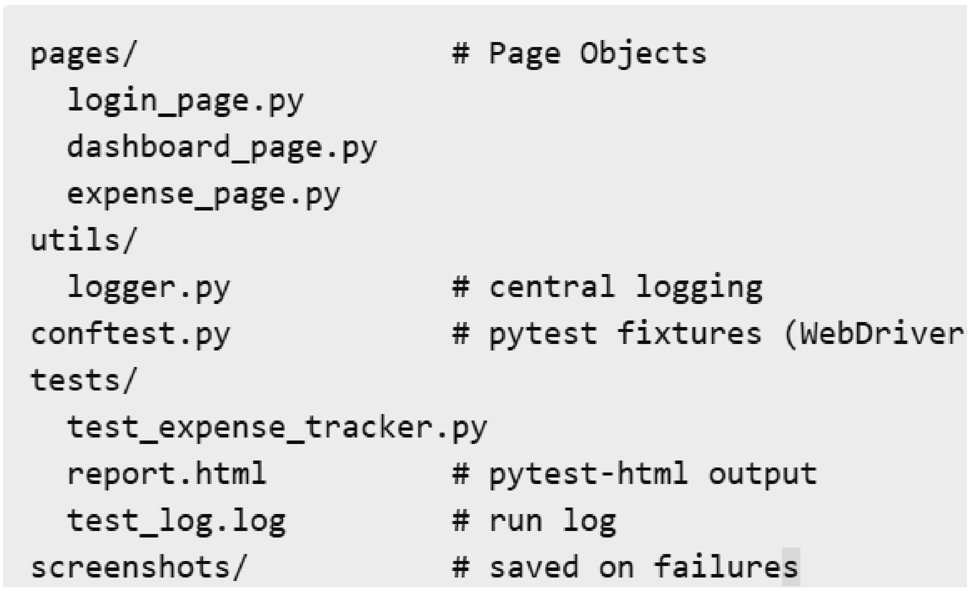
```
pytest tests/ --maxfail=1 --disable-warnings --html=tests/report.html
```

Tip: Either start a local server and use `http://127.0.0.1:8000`, or target a staging URL via `BASE_URL` environment variable.

5 Automated UI Test Suite(Selenium)

5.1 Test design and structure

This project follows the Page Object Model (POM) to keep selectors and actions separate from test logic.

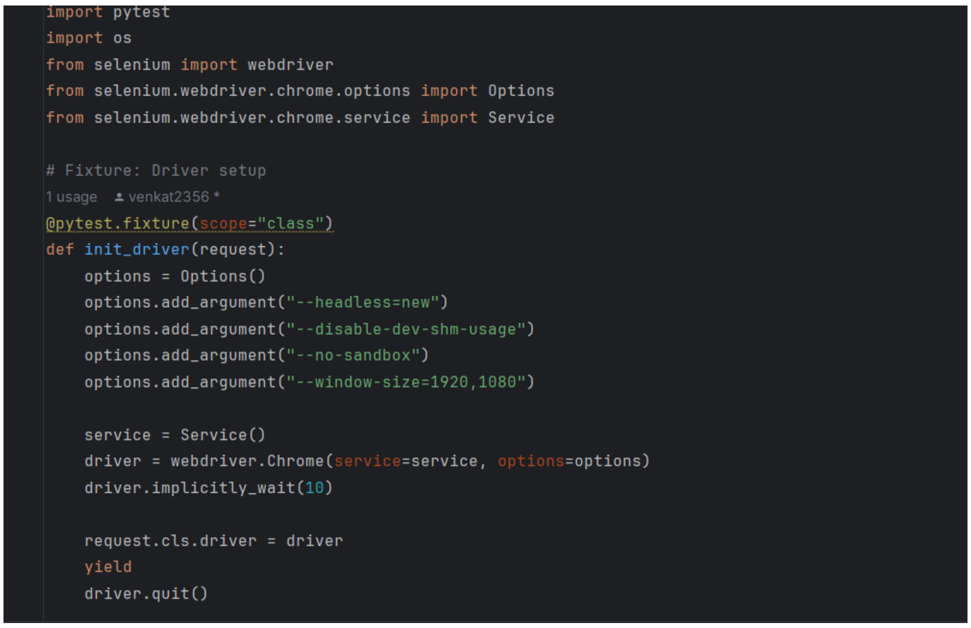


```
pages/                      # Page Objects
  login_page.py
  dashboard_page.py
  expense_page.py
utils/
  logger.py                 # central logging
conftest.py                 # pytest fixtures (WebDriver
tests/
  test_expense_tracker.py
  report.html               # pytest-html output
  test_log.log              # run log
screenshots/                # saved on failures
```

Figure 1: Selenium Automation Scripts Structure

5.2 Key fixtures (conftest.py)

The coftest.py looks like as figure 2 it icludes the driver configuration and headless mode.

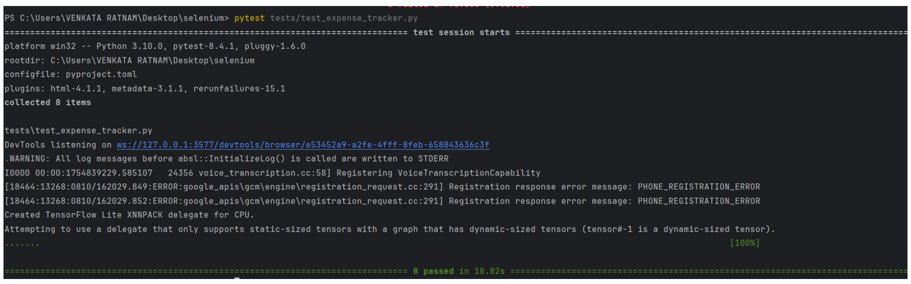


```python
import pytest
import os
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service

# Fixture: Driver setup
1 usage  venkat2356 *
@pytest.fixture(scope="class")
def init_driver(request):
    options = Options()
    options.add_argument("--headless=new")
    options.add_argument("--disable-dev-shm-usage")
    options.add_argument("--no-sandbox")
    options.add_argument("--window-size=1920,1080")

    service = Service()
    driver = webdriver.Chrome(service=service, options=options)
    driver.implicitly_wait(10)

    request.cls.driver = driver
    yield
    driver.quit()
```

Figure 2: Conftest.py

5.3 Running Locally

The test cases were run locally using the pytest command as mentioned in the section 4.6 . Running pytest locally produced 8 passed, as shown in Figure3.



```
PS C:\Users\VENKATA RATNAM\Desktop\selenium> pytest tests/test_expense_tracker.py
=============================================================================== test session starts ===============================================================================
platform win32 -- Python 3.10.0, pytest-8.4.1, pluggy-1.6.0
rootdir: C:\Users\VENKATA RATNAM\Desktop\selenium
configfile: pyproject.toml
plugins: html-4.1.1, metadata-3.1.1, rerunfailures-15.1
collected 8 items

tests\test_expense_tracker.py
DevTools listening on ws://127.0.0.1:3577/devtools/browser/e53452a9-a2fe-4fff-8feb-658843636c3f
.WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1754839229.585107   24356 voice_transcription.cc:58] Registering VoiceTranscriptionCapability
[18464:13268:0810/162029.849:ERROR:google_apis\gcm\engine\registration_request.cc:291] Registration response error message: PHONE_REGISTRATION_ERROR
[18464:13268:0810/162029.852:ERROR:google_apis\gcm\engine\registration_request.cc:291] Registration response error message: PHONE_REGISTRATION_ERROR
Created TensorFlow Lite XNNPACK delegate for CPU.
Attempting to use a delegate that only supports static-sized tensors with a graph that has dynamic-sized tensors (tensor#-1 is a dynamic-sized tensor).
.......                                                                                                                                                            [100%]

=============================================================================== 8 passed in 18.82s ================================================================================
```

Figure 3: Automation tests are passed

6 CI – test workflow

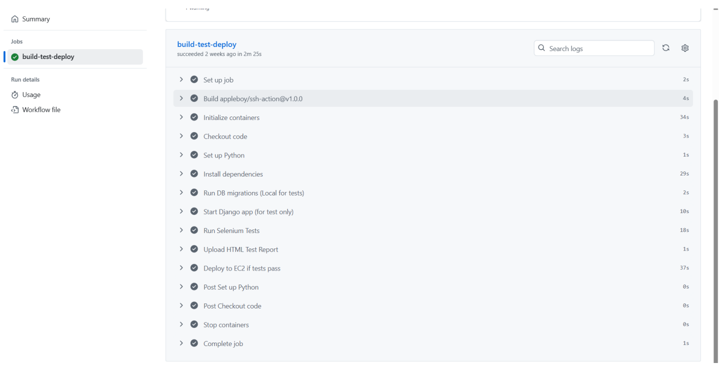
The workflow is triggered on push and pull request to master, sets up Python 3.10 and headless Chrome, then runs pytest to generate reports/pytest.html and reports/junit.xml. The actions/upload-artifact step preserves the reports so reviewers can inspect the Selenium run from the Actions page. When BASE URL is set (staging/EC2), the job skips starting a local dev server and tests against that target instead.

Workflow `.github/workflows/ci.yml`

```
name: Django CI/CD with Selenium

on:
  push:
    branches: ["master"]
  pull_request:
    branches: ["master"]

jobs:
  build-test-deploy:
    runs-on: ubuntu-latest

    services:
      selenium:
        image: selenium/standalone-chrome:latest
        ports:
          - 4444:4444
        options: --shm-size=2g

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Python
```

```
        uses: actions/setup-python@v4
        with:
          python-version: '3.10'

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install -r requirements.txt

      - name: Run DB migrations (Local for tests)
        run: |
          python manage.py makemigrations
          python manage.py migrate

      - name: Start Django app (for test only)
        run: |
          nohup python manage.py runserver 0.0.0.0:8000 >
devserver.log 2>&1 &
          sleep 10

      - name: Run Selenium Tests
        run: pytest tests/ --maxfail=1 --disable-warnings --html=
tests/report.html

      - name: Upload HTML Test Report
        uses: actions/upload-artifact@v4
        with:
          name: selenium-test-report
          path: tests/report.html
```

This run verifies the run the CI pipeline was set up to perform: the runner installed Python, dependency-managed it, ran Django migrations, started the development server, and executed the headless Selenium tests. The tests/report.html artifact was uploaded and all the steps are marked by green check which serves as the entrance to the next stage. This CI job must succeed before a step to deploy to EC2, should the tests pass, will run; the latter will be discussed in the CD section.



build-test-deploy
succeeded 2 weeks ago in 2m 25s

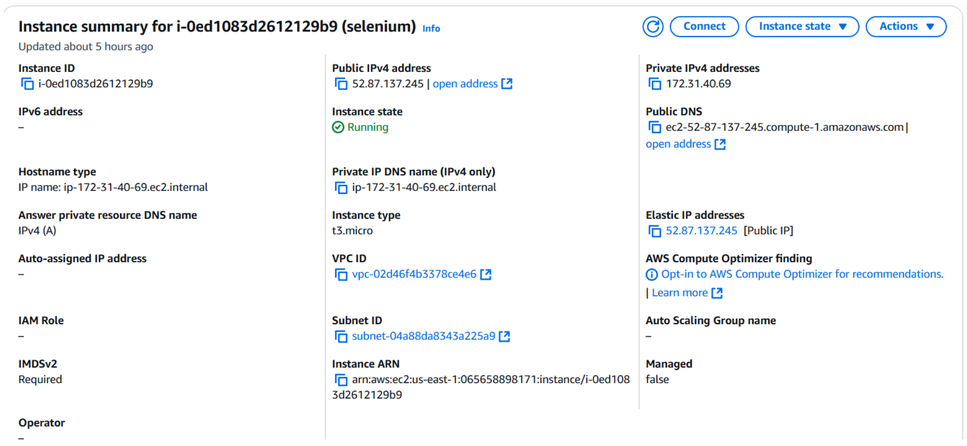
```
Set up job                              2s
Build appleboy/ssh-action@v1.0.0        4s
Initialize containers                   34s
Checkout code                           3s
Set up Python                           1s
Install dependencies                    29s
Run DB migrations (Local for tests)     2s
Start Django app (for test only)        10s
Run Selenium Tests                      18s
Upload HTML Test Report                 1s
Deploy to EC2 if tests pass             37s
Post Set up Python                      0s
Post Checkout code                      0s
Stop containers                         0s
Complete job                            1s
```

Figure 4: Pipeline Snippet

7 Cloud Provisioning (EC2) — instance, SG, user, packages, dirs

7.1 Launch an EC2 instance

The application is installed on an AWS EC2 instance; therefore, first create an EC2 instance by logging in to the AWS Management Console. The chosen configuration (AMI, instance type, key pair, and security group) is shown in Figure 5.

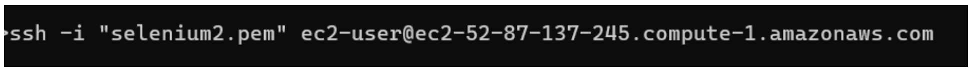


Instance summary for i-0ed1083d2612129b9 (selenium)
Updated about 5 hours ago

Instance ID i-0ed1083d2612129b9	Public IPv4 address 52.87.137.245 | open address	Private IPv4 addresses 172.31.40.69
IPv6 address –	Instance state Running	Public DNS ec2-52-87-137-245.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-40-69.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-40-69.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t3.micro	Elastic IP addresses 52.87.137.245 [Public IP]
Auto-assigned IP address –	VPC ID vpc-02d46f4b3378ce4e6	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-04a88da8343a225a9	Auto Scaling Group name –
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:065658898171:instance/i-0ed1083d2612129b9	Managed false
Operator –		

Figure 5: AWS EC2 Instance

Now connect to the created EC2 instance from the local command prompt using SSH.

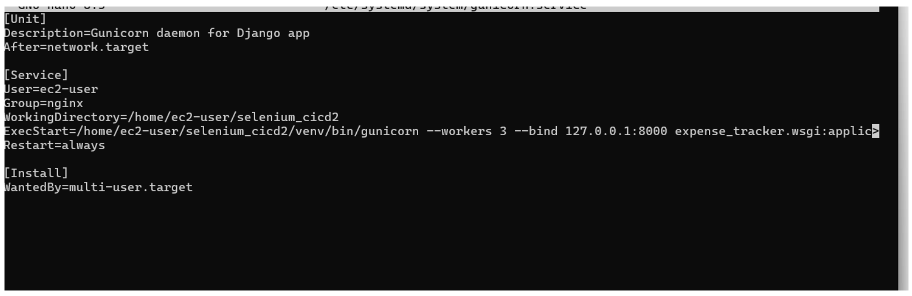


```
ssh -i "selenium2.pem" ec2-user@ec2-52-87-137-245.compute-1.amazonaws.com
```

Figure 6: SSH command to connect EC2

7.2 Gunicorn Server

Serves the Django app as a WSGI sever which has multiple worker processes that are started and managed under systemd. It does not connect to the public internet and only binds to 127.0.0.1:8000 (or a Unix socket). The Gunicorn as shown in the figure7.

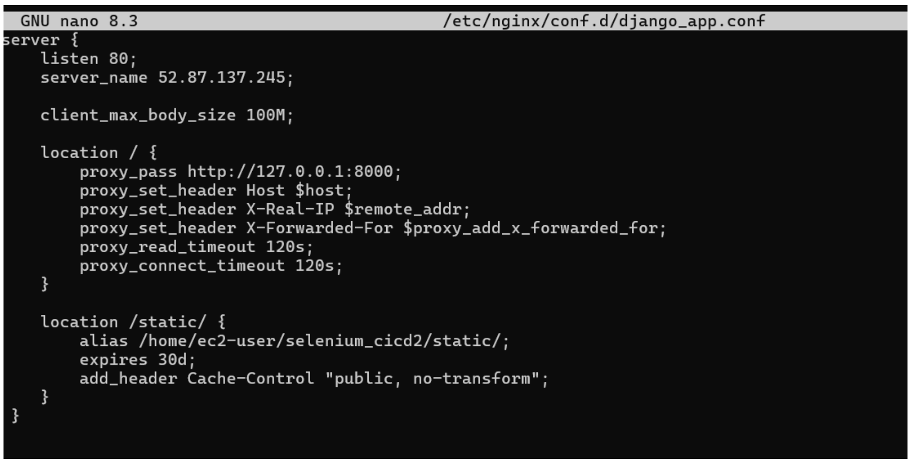


```
[Unit]
Description=Gunicorn daemon for Django app
After=network.target

[Service]
User=ec2-user
Group=nginx
WorkingDirectory=/home/ec2-user/selenium_cicd2
ExecStart=/home/ec2-user/selenium_cicd2/venv/bin/gunicorn --workers 3 --bind 127.0.0.1:8000 expense_tracker.wsgi:applic>
Restart=always

[Install]
WantedBy=multi-user.target
```

Figure 7: Gunicorn

7.3 Nginx configuration

Performs the role of the shared web server / Reverse proxy. it manages these client connections, TLS/HTTP, buffering, timeouts, etc. and serves static files, and finally delivers dynamic requests to Gunicorn.



```
 GNU nano 8.3                              /etc/nginx/conf.d/django_app.conf
server {
    listen 80;
    server_name 52.87.137.245;

    client_max_body_size 100M;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_read_timeout 120s;
        proxy_connect_timeout 120s;
    }

    location /static/ {
        alias /home/ec2-user/selenium_cicd2/static/;
        expires 30d;
        add_header Cache-Control "public, no-transform";
    }
}
```

Figure 8: Nginx server

8 CD — Deploy workflow

The CD process is shown in the below yaml and as seen it only runs after CI tests pass and it automatically deploys to EC2 via SSHing into the server, pulling the latest commit but using a read-only deploy key, rebuilding the virtual environment, executing database migrations, gathering static files, and restarting the stack (Gunicorn behind Nginx). Through GitHub Secrets secrets (EC2 host, SSH key, deploy key) are injected and they are never committed. This causes releases to be predictable and low-risk: each change comes in the same script, failures become apparent earlier, and it is easy to recover (re-run the job or roll back to a known commit).

Workflow .github/workflows/ci.yml

```
    - name: Deploy to EC2 if tests pass
        if: success()
        uses: appleboy/ssh-action@v1.0.0
        with:
          host: ${{ secrets.EC2_HOST }}
          username: ec2-user
          key: ${{ secrets.EC2_SSH_KEY }}

          script_before: |
            mkdir -p ~/.ssh
            chmod 700 ~/.ssh
            echo "${{ secrets.SSH_DEPLOY_KEY_GITHUB }}" | tee ~/.
  ssh/id_rsa_github_deploy > /dev/null
            chmod 600 ~/.ssh/id_rsa_github_deploy
            ssh-keyscan github.com >> ~/.ssh/known_hosts 2>/dev/
   null
            chmod 644 ~/.ssh/known_hosts
            echo "Host github.com" > ~/.ssh/config
            echo "  IdentityFile ~/.ssh/id_rsa_github_deploy" >>
   ~/.ssh/config
            echo "  StrictHostKeyChecking no" >> ~/.ssh/config
            chmod 600 ~/.ssh/config

          script: |
            sudo yum install -y git
            cd ~
            if [ ! -d selenium_cicd2 ]; then
              git clone git@github.com:venkat2356/selenium_cicd2.
   git
            fi
            cd selenium_cicd2

            echo "Pulling latest code..."
            git reset --hard
            git clean -fd
            git pull origin master

            echo "Removing Python cache..."
            find . -type d -name "__pycache__" -exec rm -rf {} +
            find . -name "*.pyc" -delete

            echo "Rebuilding virtual environment..."
            python3 -m venv venv
            source venv/bin/activate
            pip install --upgrade pip
            pip install -r requirements.txt

            echo "Applying DB migrations..."
            python manage.py migrate

            echo "Collecting static files..."
            python manage.py collectstatic --noinput

            echo "Setting up Gunicorn as a service..."
            echo "[Unit]
            Description=Gunicorn daemon for Django app
            After=network.target

            [Service]
            User=ec2-user
            Group=nginx
            WorkingDirectory=/home/ec2-user/selenium_cicd2
            ExecStart=/home/ec2-user/selenium_cicd2/venv/bin/
   gunicorn --workers 3 --bind 127.0.0.1:8000 expense_tracker.
```

```
wsgi:application
         Restart=always

         [Install]
         WantedBy=multi-user.target" | sudo tee /etc/systemd/
system/gunicorn.service > /dev/null

         sudo systemctl daemon-reexec
         sudo systemctl daemon-reload
         sudo systemctl enable gunicorn
         sudo systemctl restart gunicorn
         sudo systemctl status gunicorn --no-pager

         echo "Restarting Nginx..."
         sudo nginx -t
         sudo systemctl restart nginx

         echo "Deployment complete."
```

GitHub Secrets for SSH-based CD

Store the connection details and keys in **GitHub Secrets** (Repository → Settings → Secrets and variables → Actions). Create the following:

Name	**What to store**	**Used by**
`EC2_HOST`	EC2 public IP or DNS (e.g., `52.87.137.245` or `ec2-52-87-137-245.compute-1.amazonaws.com`).	`ssh-action` host
`EC2_SSH_KEY`	*Full* PEM private key used for SSH login: paste from `---BEGIN OPENSSH PRIVATE KEY---` to the closing line, preserving line breaks.	`ssh-action` key
`SSH_DEPLOY_KEY_GITHUB`	Read-only deploy key for cloning the private repo on the server (store the *private* half here; add the *public* half as a Deploy Key in GitHub).	SSH config in `script_before`

Table 1: GitHub Secrets for SSH-based CD.

These secrets are referenced in the workflow as:

```
with:
  host: ${{ secrets.EC2_HOST }}
  username: ec2-user
  key: ${{ secrets.EC2_SSH_KEY }}

# Deploy key used inside script_before
echo "${{ secrets.SSH_DEPLOY_KEY_GITHUB }}" > ~/.ssh/id_rsa_github_deploy
```

GitHub `Settings → Secrets and variables → Actions` showing `EC2_HOST`, `EC2_SSH_KEY`, `SSH_DEPLOY_KEY_GITHUB`.

9 First deployment & verification

Figure 9 indicates that the home page of the Expense Tracker loads correctly on the EC2 instance at the URL http://52.87.137.245/ after creating an EC2 instance, installing NGINX and Gunicorn, and running application. The asset-built landing page itself and the assets it contains verify that Nginx is responding to requests, Gunicorn is serving the Django application and the static files metadata was correctly compiled- fulfilling the first-deployment check.

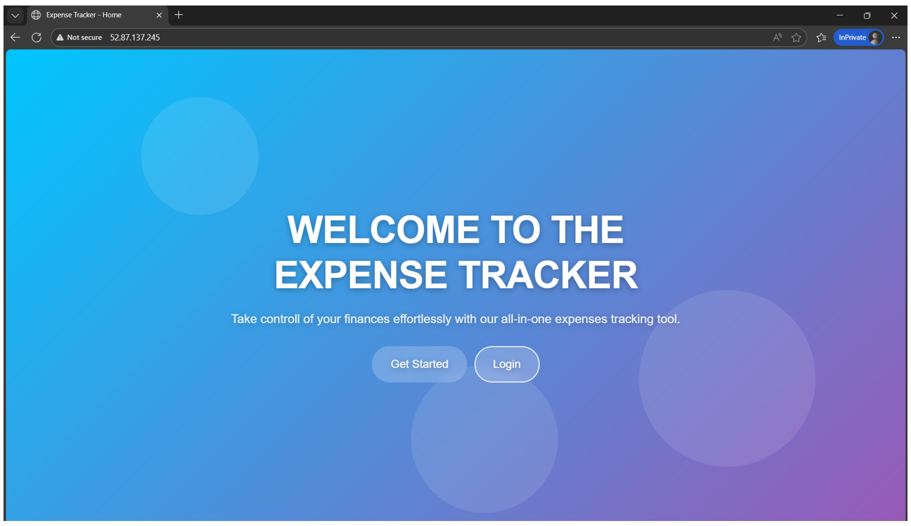

Expense Tracker - Home
Not secure 52.87.137.245
InPrivate
WELCOME TO THE EXPENSE TRACKER
Take controll of your finances effortlessly with our all-in-one expenses tracking tool.
Get Started
Login

Figure 9: Deployed Application

References

Amazon Web Services (2025). *Amazon EC2 User Guide — Instances.*
URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Instances.html.

Django Software Foundation (2025). *Deploying Django — Deployment checklist.*
URL: https://docs.djangoproject.com/en/stable/howto/deployment/checklist/.

GitHub (2025). *GitHub Actions — Workflow syntax.*
URL: https://docs.github.com/en/actions/using-workflows/workflow-syntax-for-github-actions.

NGINX, Inc. (2025). *NGINX HTTP proxying (reverse proxy).*
URL: https://nginx.org/en/docs/http/ngx_http_proxy_module.html.

pytest-dev (2025a). *pytest Documentation.*
URL: https://docs.pytest.org/en/stable/.

— (2025b). *pytest-html — HTML reporting plugin.*
URL: https://pytest-html.readthedocs.io/.

SeleniumHQ (2025). *Selenium Documentation — WebDriver.*
URL: https://www.selenium.dev/documentation/webdriver/.