

Online Event Calendar

Cloud Computing Project Final Report

Shubham Narandekar

A report submitted in part fulfilment of the Cloud Computing module of
MSc Computer Science.



School of Computer Science and Informatics

University College Dublin

30 December 2020

Abstract

Cloud Computing is becoming emerging platform to allow users build, store and deploy their applications into cloud server on a Pay-as-you-Go service. Users can scale these on-demand services as per their needs. Just like Google Calendar, this project is a web application that allows user to: -

1. Create Events
2. Edit and Delete existing events
3. Search for events
4. Time slot booking
5. Share calendar

This web application is then deployed on Amazon EC2 Linux instance so that it can be accessed from anywhere anytime using the public dns of the instance.

Table of Contents

1	Introduction	3
2	Project's Specifications and Requirements.....	3
3	Methodology & System Architecture.....	5
4	Implementation Details.....	6
5	Conclusion	9
6	References.....	9

1 Introduction

- Motivation: -
 - To allow users to get a handy online event calendar.
 - To get hands on experience in full stack development.
 - To understand the process of deploying a web application on cloud.
- Project objectives: -
 - To allow multiple users to register to the application.
 - To allow users to maintain their own specific calendar.
 - To allow users to keep track of their events.
 - To allow users to make some changes to their events or delete the event if required.
 - To allow users to share their events with their friends.
- This project is interesting because during the implementation I got to learn many different concepts. I gained knowledge of the flow in which the entire frontend and backend processes work. I got to know about various calendar grid API provided by Django. The most interesting part was the deployment of the application on AWS EC2.
- We can implement this application on cloud computing by deploying it on various servers provided by various cloud providers. For instance, this application can be deployed on Amazon's Elastic Compute Cloud instance with Linux as an operating system. This will allow users to access the application from anywhere and anytime until and unless the EC2 instance is running. This is useful because the developer doesn't have to take care of the server maintenance and reliability. Everything is managed by the cloud service provider.

2 Project's Specifications and Requirements

- Specifications: -
 - A user can register to the application by providing his details and creating an account on the register's page.

The screenshot shows a web browser window with the URL `ec2-34-244-11-148.eu-west-1.compute.amazonaws.com/formpage/register`. The page has a dark header with the text "Event Calender" and links for "Log in" and "About the project". A blue "Register" button is in the top right corner. The main content area is titled "Enter Your details" and contains a registration form with the following fields and labels:

- Username***: A text input field with a small "i" icon. Below it, a note says "Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only."
- Email***: A text input field.
- Firstname***: A text input field.
- Lastname***: A text input field.
- Password***: A text input field. Below it, four bullet points list password requirements:
 - Your password can't be too similar to your other personal information.
 - Your password must contain at least 6 characters.
 - Your password can't be a commonly used password.
 - Your password can't be entirely numeric.
- Password confirmation***: A text input field. Below it, a note says "Enter the same password as before, for verification."

A blue "Register" button is located at the bottom of the form.

- After registering, user can log in to access the calendar. The calendar will be user specific and can only be accessed by that specific user.

The screenshot shows a web browser window with the URL `ec2-34-244-11-148.eu-west-1.compute.amazonaws.com/homepage/login`. The page has a dark header with 'Event Calendar' on the left, 'Log in' and 'About the project' in the center, and a 'Register' button on the right. The main content area is titled 'Provide Credentials' and contains a login form with two input fields: 'Username*' and 'Password*'. Below the fields is a blue 'Login' button.

- Once the user is logged in, he/she can go through the calendar to check for an upcoming event. Also, user can jump between the months as required.

The screenshot shows the calendar view for December 2020. The header includes 'There you go!, 20200132' on the left, a 'Months' dropdown menu in the center, and 'Add Event' and 'Logout' buttons on the right. The calendar grid has columns for days of the week (Mon to Sun) and rows for dates. A single event titled 'Hello' is scheduled for December 24th at 10:00 AM.

December 2020						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24 • Hello	25	26	27

- User can then add an event by clicking the Add Event tab on the navigation bar. It will give user the following form to add the details of the event.

The screenshot shows the 'Add Event' form. The header includes 'There you go!, 20200132' on the left, a 'Logout' button on the right, and a 'Calendar' button in the top right corner. The form has the following fields: 'Title*' (text input), 'Description*' (text area), 'Start time*' (datetime picker with format 'dd-mm-yyyy --:--'), and 'End time*' (datetime picker with format 'dd-mm-yyyy --:--'). A blue 'Submit' button is at the bottom.

- Once an event is created, user can perform various operations on that event like sharing with other users, editing or deleting the existing user, editing the time slots of the event.

The screenshot shows a web browser window with the URL `ec2-34-244-11-148.eu-west-1.compute.amazonaws.com/calendar/event/11/details/`. The page has a dark header with the text "There you go!, 20200132" and a "Logout" link. Below the header, there are buttons for "Delete Event", "Edit Event", and "Calendar". The main content area displays the event details: "Event Name: hello", "From Dec. 25, 2020, 12:33 p.m. To Dec. 28, 2020, 12:33 p.m.", "Description: Wassup?", and "Shared with :-". A "Share" button is located to the right of the "Shared with :-" text. Below this, there is a table with two columns: "#", "Name", and "Action". The table contains one row with the value "1" in the "#", "user1" in the "Name", and a "Remove" button in the "Action" column.

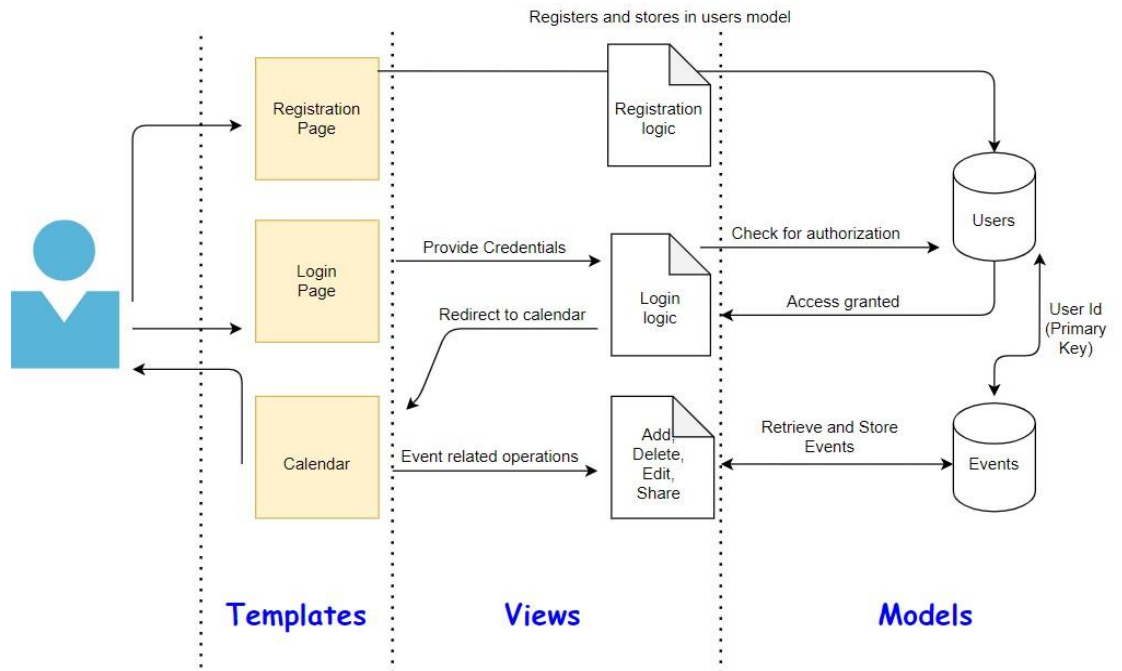
#	Name	Action
1	user1	Remove

- Requirements:
 1. Software requirements:
 - Operating System - Windows/Linux
 - Front End - HTML5, CSS, Bootstrap
 - Back End - Python 3.6 or above
 - Framework - Django
 - Database - SQLite
 - IDE - VS Code
 2. Hardware requirements:
 - OS - 64-bit
 - RAM - 1GB (min)
 - Processor - i3 (min)
 - Hard Disk - 20GB

3 Methodology & System Architecture

- Methodology:
 - The main aim of this project is to provide users with an online event calendar to keep track and notify them about their day-to-day events. Django provides a user model where all the data related to a user and the users who register with the web application is stored in the user model. This model can be used to link each event with a unique user id which is the primary key of the user model. This helps to uniquely differentiate between different user specific events. So, whenever a user creates an event, the user id of that user is assigned to that particular event and further that user can perform all the operations on that event.

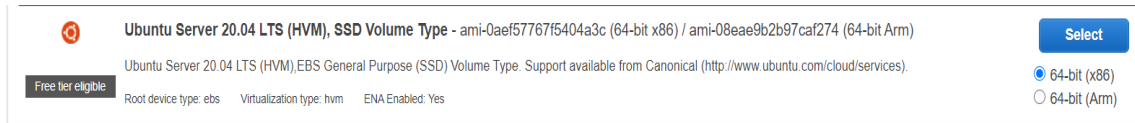
- System architecture:
 - I have used Django framework to develop the web application which is in mainly three parts i.e., Model, Views and Templates.
 - Following figure illustrates the architecture of the proposed web application.



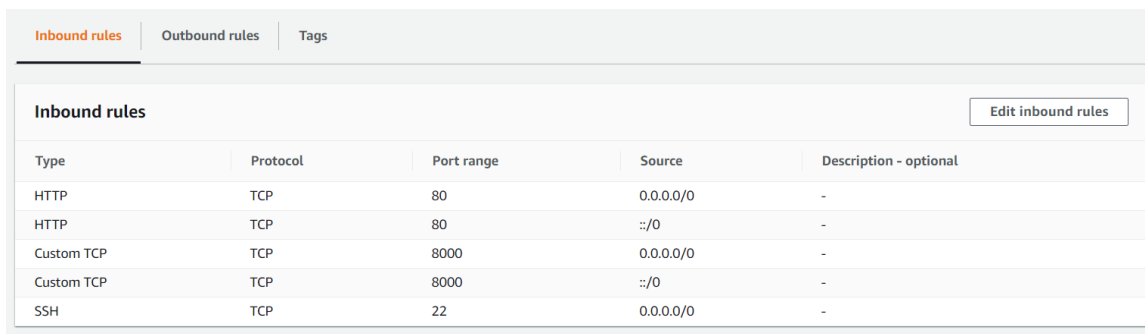
4 Implementation Details

- Tools and technologies used:
 - Framework: Django 2.2.10
 - Language: Python 3.9.0
 - Frontend: HTML, Bootstrap and CSS
 - Database: SQLite (Handled by Django)
- Functions and operations involved in the application are:
 - Add event: Allows user to add event with a starting and ending date along with time.
 - Delete event: Allows user to delete the selected event if no longer needed.
 - Edit event: Allows user to make changes to an existing event.
 - Share event: Allows user to share the selected event with other users.
- Steps to install and run the project on localhost:
 - Unzip the project folder.
 - cd into the project folder.
 - install python 3.6 or above if it's not installed.
 - **\$pip install virtualenv** (or pip install pipenv)

- **\$virtualenv cloud** (create new virtual environment)
- **\$cloud/Scripts/activate** (Activate your newly created environment)
- **\$pip install -r requirements.txt** (To install Django and other packages)
- **\$python manage.py runserver**
- To deploy the applications:
 - Login to AWS console and launch and EC2 instance:



- Review and launch your instance and create and download a KeyPair.pem file.
- Open your settings.py from your project folder and do the following changes:
 1. `DEBUG = False`
 2. `ALLOWED_HOSTS = ["*"]`
 3. `LOGIN_REDIRECT_URL = '<PASTE PUBLIC DNS OF YOUR INSTANCE>/calender/'`
For example: `LOGIN_REDIRECT_URL = 'http://ec2-34-244-11-148.eu-west-1.compute.amazonaws.com/calender/'`
- After the above changes save the file and create a git repository and upload your entire project in that repository.
- Once instance is running, open your cmd and cd into the directory where you have downloaded the KeyPair.pem. After that follow the commands below.
- **\$chmod 400 KeyPair.pem**
- **\$ssh -i "First EC2 KeyPair.pem" ubuntu@<Paste your instance's Public DNS>**
- For example: **ssh -i "KeyPair.pem" ubuntu@ec2-34-244-11-148.eu-west-1.compute.amazonaws.com**
- Now that you are connected to your instance, edit your security group and add the following inbound rules:



Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
HTTP	TCP	80	::/0	-
Custom TCP	TCP	8000	0.0.0.0/0	-
Custom TCP	TCP	8000	::/0	-
SSH	TCP	22	0.0.0.0/0	-

- After doing ssh into your instance follow the following commands.
 1. **\$sudo apt-get update**

2. **\$sudo apt-get upgrade -y**
3. **\$sudo apt-get install python3-venv**
4. **\$python3 -m venv cloud**
5. **\$source cloud/bin/activate**
6. **\$git clone <PASTE URL OF YOUR PROJECT REPOSITORY>**
7. **\$cd into your project folder.**
8. **\$pip3 install -r requirements.txt**
9. **\$pip3 install gunicorn**
10. **\$sudo apt-get install -y nginx**
11. **\$sudo apt-get install -y supervisor**
12. **\$cd /etc/supervisor/conf.d/**
13. **\$sudo touch gunicorn.conf**
14. **\$sudo nano gunicorn.conf** (Open the file and add the following configuration to it and save it)

```
ubuntu@ip-172-31-32-251: /etc/supervisor/conf.d
GNU nano 2.9.3 gunicorn.conf

[program:gunicorn]
directory=/home/ubuntu/DjangoWebApplication
command=/home/ubuntu/cloud/bin/gunicorn --workers 3 --bind unix:/home/ubuntu/DjangoWebApplication/app.sock project.wsgi:application
autostart=true
autorestart=true
stderr_logfile=/var/log/gunicorn/gunicorn.err.log
stdout_logfile=/var/log/gunicorn/gunicorn.out.log

[group:gunicorn]
programs:gunicorn
```

15. **\$sudo mkdir /var/log/gunicorn**
16. **\$sudo supervisorctl reread**
17. **\$sudo supervisorctl update**
18. **\$sudo supervisorctl status**
19. **\$cd (into home directory)**
20. **\$cd /etc/nginx/sites-available/**
21. **\$sudo touch Django.conf**
22. **\$sudo nano Django.conf** (Open the file and add the following configurations to it and save it)

```
GNU nano 2.9.3 django.conf Modified

server {
    listen 80;
    server_name ec2-34-244-11-148.eu-west-1.compute.amazonaws.com;

    location / {
        include proxy_params;
        proxy_pass http://unix:/home/ubuntu/DjangoWebApplication/app.sock;
    }
    location /static/ {
        autoindex on;
        alias /home/ubuntu/DjangoWebApplication/static/;
    }
}
```


23. `$sudo nginx -t`

24. `$echo "server_names_hash_bucket_size 128;" | sudo tee
/etc/nginx/conf.d/server_names_hash_bucket_size.conf`

25. `$sudo ln Django.conf /etc/nginx/sites-enabled/`

26. `$sudo nginx -t`

27. `$sudo service nginx restart`

- Once you are done with the above process, the web application will be successfully deployed.
- You can access the application using the following link:
<Public DNS of your instance>/homepage/
For example: **http://ec2-34-244-11-148.eu-west-1.compute.amazonaws.com/homepage/**
- Following is the link to my application after deployment is done:

<http://ec2-54-229-192-162.eu-west-1.compute.amazonaws.com/homepage/>

- WebApplicationTesting Username: **demo**
 Password: **testing@123**
- AWS account details (IAM) AccountID: **shubhamnarandekar**
 Username: **projecttesting**
 Password: **projecttesting@123**

5 Conclusion

- I have successfully achieved the goal of developing an online event calendar web application and deploying it on the cloud. I have also gained hands on experience on full stack development where I got to learn and use Django framework which is a widely used framework for developing web applications. There were couple of challenges that I faced where my application wasn't able to access the static (images, CSS) files of the project after deployment. I managed to resolve this issue by configuring the nginx configuration to read my static files from a specified path. In this way, I successfully the deployed the application and learned all the necessary steps that needs to be taken to deploy an application on cloud. Also, I have learned the entire flow in which the Django framework works. It follows Model-View-Template when used with python whereas all other frameworks follow Model-View-Controller.

6. References

- https://www.youtube.com/watch?v=dy1oG_XqsgY&ab_channel=SajibHossain
- https://www.youtube.com/watch?v=u0oElqQV-E&ab_channel=ShobiPP