**Cloud Server Project**

Global IP address: 54.172.7.187

DNS: www.samonshahzad.com

Name: Samon Shahzad

Student ID: 35562778

[Introduction 1](#_Toc519505649)

[Project Overview 2](#_Toc2007302880)

[Setting up a Web Server 3](#_Toc120652478)

[Launching an AWS EC2 Instance 3](#_Toc1572134820)

[Setting up SSH Access 3](#_Toc1866949666)

[Initial Server Setup 5](#_Toc27298285)

[Setting up Nginx 5](#_Toc2022977985)

[Configuring the Flask Application 6](#_Toc767985243)

[Cloning the Application Repository 6](#_Toc561000957)

[Setting up a Virtual Environment 6](#_Toc1573297088)

[Creating a WSGI Entry Point 6](#_Toc1111920521)

[Configuring Nginx as a Reverse Proxy 6](#_Toc193942260)

[Creating a Systemd Service File 7](#_Toc1621923007)

[Setting up the Database 8](#_Toc1695753199)

[Linking with a DNS Entry 9](#_Toc1339603702)

[Domain Registration with Namecheap 9](#_Toc981270746)

[Setting up DNS Records 9](#_Toc1612243054)

[SSL/TLS Configuration 9](#_Toc1769422513)

[Custom Scripts 10](#_Toc705440090)

[Application Monitoring and Maintenance Script 10](#_Toc1968952878)

[Functions of the script 13](#_Toc1568724841)

[Server Testing and Validation 15](#_Toc1298778470)

[Verifying Nginx Configuration 15](#_Toc1087437495)

[Verifying SSL Configuration 15](#_Toc1931796148)

[Testing the Flask Application 15](#_Toc1638509782)

[Testing Database Connection 15](#_Toc1449822657)

[Screenshots of Working Application 16](#_Toc1883621804)

[Maintenance Documentation 17](#_Toc1407803028)

[Regular Updates 17](#_Toc322074931)

[Application Updates 17](#_Toc2075420389)

[Database Maintenance 17](#_Toc1136592596)

[Nginx and SSL Maintenance 18](#_Toc841580580)

[Cost Analysis 18](#_Toc1935239621)

[Cost Optimization Strategies 18](#_Toc1257245074)

[Troubleshooting Common Issues 19](#_Toc266231753)

[Nginx 502 Bad Gateway Error 19](#_Toc1654705639)

[Database Connection Issues 19](#_Toc1284543359)

[SSL Certificate Issues 20](#_Toc141726090)

[Application Loading Errors 20](#_Toc275207234)

[References 21](#_Toc1845544317)

## Introduction

This document outlines the complete setup and configuration of a cloud-based web server using AWS EC2, with Nginx as a reverse proxy for a Flask application. The server is accessible via a custom domain name (www.samonshahzad.com) with secure HTTPS access. This documentation is designed to allow replication of the server environment from scratch if needed, and includes all necessary commands, configurations, and explanations.

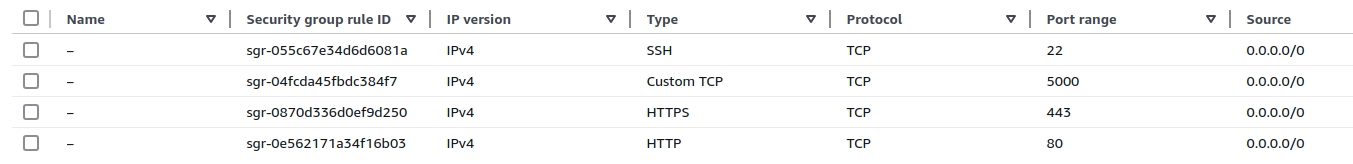
### Project Overview

The deployed application is an event planning and management platform called “samoonEvents”. It allows users to create and manage events, handle guest lists and invitations, coordinate with vendors, track tasks, and manage budgets. The application demonstrates a complete web stack with a Flask backend, SQLAlchemy for database operations, and a responsive frontend using Tailwind CSS, all hosted on AWS infrastructure.

## Setting up a Web Server

### Launching an AWS EC2 Instance

1. Sign in to AWS Management Console and navigate to EC2 service
2. Click “Launch Instance”
3. Select Ubuntu Server 22.04 LTS
4. Choose an instance type (t2.micro is eligible for free tier)
5. Configure security groups to allow traffic:
   * SSH (Port 22) from your IP
   * HTTP (Port 80) from anywhere
   * HTTPS (Port 443) from anywhere
   * Custom TCP (Port 5000) from anywhere (for Flask development)



### Setting up SSH Access

Generate an SSH key pair during instance creation or use an existing one. After downloading the .pem file, set the correct permissions:

mv samon-key.pem ~/.ssh/  
 chmod 600 ~/.ssh/samon-key.pem

Create an SSH config file for easier access:

vim ~/.ssh/config

Add the following configuration:

Host samoon-server  
 HostName 54.172.7.187  
 User ubuntu  
 IdentityFile ~/.ssh/samon-key.pem

Now you can connect with:

ssh samoon-server

### Initial Server Setup

Update the system packages:

sudo apt update  
sudo apt upgrade -y

## 

For AWS instances, they come with some essential packages installed such as python, git, mysql, etc.

## Setting up Nginx

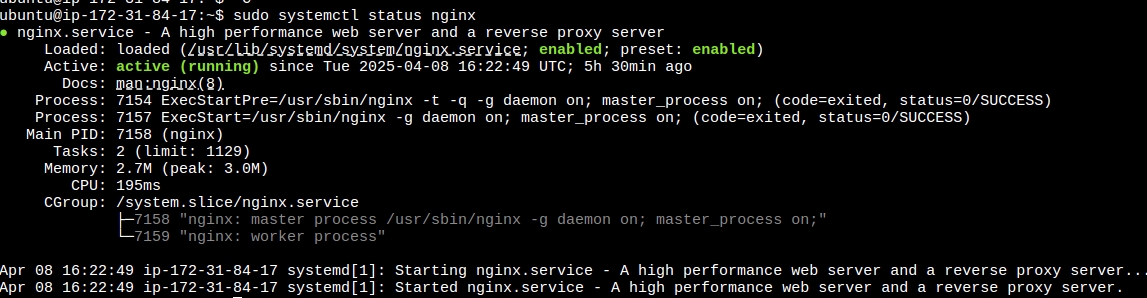
Install Nginx:

sudo apt install nginx -y

Verify Nginx is running:

sudo systemctl status nginx

You should see output indicating that Nginx is active (running):



## Configuring the Flask Application

### Cloning the Application Repository

cd ~  
git clone https://github.com/ngari-kev/samoonEvents.git  
cd samoonEvents

### Setting up a Virtual Environment

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

Install dependencies:

pip install -r requirements.txt  
pip install gunicorn

### Creating a WSGI Entry Point

The application already includes a wsgi.py file with the following code:

from samoonEvents.app import app  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run()

### 

### Configuring Nginx as a Reverse Proxy

Create a new Nginx configuration file:

sudo vim /etc/nginx/sites-available/samoonevents

Add the following configuration:

server {  
 listen 80;  
 server\_name 54.172.7.187 www.samonshahzad.com;  
  
 location /static {  
 alias /home/ubuntu/samoonEvents/samoonEvents/static;  
 }  
  
 location / {  
 proxy\_pass http://localhost:5000;  
 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\_set\_header X-Forwarded-Proto $scheme;  
 }  
}

Enable the configuration by creating a symbolic link:

sudo ln -s /etc/nginx/sites-available/samoonevents /etc/nginx/sites-enabled

Test Nginx configuration:

sudo nginx -t

If the test is successful, restart Nginx:

sudo systemctl restart nginx

### 

### Creating a Systemd Service File

Create a service file to automatically start the Flask application with Gunicorn:

sudo vim /etc/systemd/system/samoonevents.service

Add the following configuration:

[Unit]  
Description=Gunicorn instance to serve samoonEvents Flask application  
After=network.target  
  
[Service]  
User=ubuntu  
WorkingDirectory=/home/ubuntu/samoonEvents  
Environment="PATH=/home/ubuntu/samoonEvents/.venv/bin"  
ExecStart=/home/ubuntu/samoonEvents/.venv/bin/gunicorn --workers 3 --bind 0.0.0.0:5000 wsgi:app  
Restart=always  
  
[Install]  
WantedBy=multi-user.target

Start and enable the service:

sudo systemctl start samoonevents  
sudo systemctl enable samoonevents  
sudo systemctl status samoonevents

## Setting up the Database

Initialize and create the database:

cd ~/samoonEvents  
source .venv/bin/activate  
export FLASK\_APP=samoonEvents.app  
flask db init  
flask db migrate -m "Initial migration"  
flask db upgrade

Populate the database with sample data:

python -c samoonEvents.generate\_data

## Linking with a DNS Entry

### Domain Registration with Namecheap

1. Register a domain with Namecheap (www.samonshahzad.com)
2. Navigate to the Domain List and select “Manage”
3. Go to the “Advanced DNS” tab

### Setting up DNS Records

Create an A Record that points to your EC2 instance:

* Type: A Record
* Host: @ (for root domain)
* Value: 54.172.7.187 (your EC2 IP address)
* TTL: 30 minutes

Create an A Record for www subdomain:

* Type: A Record
* Host: www
* Value: 54.172.7.187 (your EC2 IP address)
* TTL: 30 minutes

Allow time for DNS propagation (typically 30 minutes to 48 hours).

## 

## SSL/TLS Configuration

Install Certbot:

sudo apt install certbot python3-certbot-nginx -y

Obtain and install SSL certificate:

sudo certbot --nginx -d www.samonshahzad.com -d samonshahzad.com

Follow the prompts, providing your email and accepting the terms of service.

Certbot will automatically update your Nginx configuration to redirect HTTP traffic to HTTPS and set up the SSL certificate.

Verify the automatic renewal:

sudo certbot renew --dry-run

## Custom Scripts

### Application Monitoring and Maintenance Script

Below is a custom Bash script that performs regular monitoring and maintenance of the application, including backups, log rotation, and health checks.

Create a file named maintain\_samoonevents.sh in your home directory:

vim ~/maintain\_samoonevents.sh

Add the following script:

#!/bin/bash  
  
# samoonEvents Maintenance Script  
# This script performs routine maintenance tasks for the samoonEvents application  
# Author: Samon Shahzad  
# Created: April 2025  
  
# Configuration  
APP\_DIR="/home/ubuntu/samoonEvents"  
BACKUP\_DIR="/home/ubuntu/backups"  
LOG\_DIR="/home/ubuntu/logs"  
DATE=$(date +%Y-%m-%d\_%H-%M-%S)  
BACKUP\_FILENAME="samoonevents\_backup\_$DATE.tar.gz"  
MAIN\_LOG="$LOG\_DIR/maintenance\_log.txt"  
APP\_DB="$APP\_DIR/samoonEvents/app.db"  
EMAIL="35562778@student.murdoch.edu.au"  
  
# Create directories if they don't exist  
mkdir -p "$BACKUP\_DIR" "$LOG\_DIR"  
  
# Log function  
log() {  
 echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$MAIN\_LOG"  
}  
  
log "Starting maintenance tasks for samoonEvents"  
  
# Check if application service is running  
if systemctl is-active --quiet samoonevents; then  
 log "Service check: samoonevents service is running"  
else  
 log "WARNING: samoonevents service is not running! Attempting to restart..."  
 sudo systemctl restart samoonevents  
  
 # Check if restart was successful  
 if systemctl is-active --quiet samoonevents; then  
 log "Service recovery: Successfully restarted samoonevents service"  
 else  
 log "ERROR: Failed to restart samoonevents service!"  
 echo "samoonEvents service is down and could not be restarted!" | mail -s "ALERT: samoonEvents Service Down" "$EMAIL"  
 fi  
fi  
  
# Perform database backup  
log "Starting database backup..."  
if [ -f "$APP\_DB" ]; then  
 # Create backup directory for this date if it doesn't exist  
 mkdir -p "$BACKUP\_DIR/db\_backups"  
  
 # Copy the SQLite database file  
 cp "$APP\_DB" "$BACKUP\_DIR/db\_backups/app\_db\_$DATE.sqlite"  
  
 if [ $? -eq 0 ]; then  
 log "Database backup successful: app\_db\_$DATE.sqlite"  
 else  
 log "ERROR: Database backup failed!"  
 fi  
else  
 log "ERROR: Database file not found at $APP\_DB"  
fi  
  
# Create application backup  
log "Creating application backup..."  
tar -czf "$BACKUP\_DIR/$BACKUP\_FILENAME" "$APP\_DIR" 2>/dev/null  
  
if [ $? -eq 0 ]; then  
 log "Application backup successful: $BACKUP\_FILENAME"  
  
 # Clean up old backups (keep only the 7 most recent)  
 log "Cleaning up old backups..."  
 ls -t "$BACKUP\_DIR"/\*.tar.gz | tail -n +8 | xargs -r rm  
 log "Retained most recent 7 backups"  
else  
 log "ERROR: Application backup failed!"  
fi  
  
# Check disk space  
DISK\_USAGE=$(df -h / | awk 'NR==2 {print $5}' | sed 's/%//')  
if [ "$DISK\_USAGE" -gt 85 ]; then  
 log "WARNING: Disk space is critical: ${DISK\_USAGE}%"  
 echo "Disk space on samoonEvents server is at ${DISK\_USAGE}%" | mail -s "ALERT: Low Disk Space on samoonEvents Server" "$EMAIL"  
else  
 log "Disk space check: ${DISK\_USAGE}% used (OK)"  
fi  
  
# Check for system updates  
log "Checking for system updates..."  
sudo apt update &>/dev/null  
UPDATES=$(apt list --upgradable 2>/dev/null | grep -c "upgradable")  
SECURITY\_UPDATES=$(apt list --upgradable 2>/dev/null | grep -c "security")  
  
if [ "$SECURITY\_UPDATES" -gt 0 ]; then  
 log "WARNING: $SECURITY\_UPDATES security updates available!"  
 echo "$SECURITY\_UPDATES security updates are available on the samoonEvents server!" | mail -s "ALERT: Security Updates Available" "$EMAIL"  
else  
 log "Security check: No security updates required"  
fi  
  
if [ "$UPDATES" -gt 0 ]; then  
 log "System has $UPDATES packages that can be upgraded"  
else  
 log "System is up to date"  
fi  
  
# Rotate logs if they're getting too large  
find "$LOG\_DIR" -type f -name "\*.log" -size +100M | while read log\_file; do  
 log "Rotating large log file: $log\_file"  
 mv "$log\_file" "${log\_file}.1"  
 touch "$log\_file"  
done  
  
# Perform HTTP health check  
HTTP\_STATUS=$(curl -s -o /dev/null -w "%{http\_code}" https://www.samonshahzad.com)  
if [ "$HTTP\_STATUS" -eq 200 ]; then  
 log "Website health check: HTTP Status $HTTP\_STATUS (OK)"  
else  
 log "WARNING: Website health check failed! HTTP Status: $HTTP\_STATUS"  
 echo "Website health check failed with HTTP status $HTTP\_STATUS" | mail -s "ALERT: samoonEvents Website Down" "$EMAIL"  
fi  
  
log "Maintenance tasks completed for samoonEvents"  
log "---------------------------------------------"

Make the script executable:

chmod +x ~/maintain\_samoonevents.sh

Set up a cron job to run the script daily:

crontab -e

Add the following line to run the maintenance script at 3 AM every day:

0 3 \* \* \* /home/ubuntu/maintain\_samoonevents.sh

### Functions of the script

This maintenance script performs several key functions:

1. Checks if the application service is running and attempts to restart it if not
2. Creates timestamped backups of both the application files and database
3. Implements a retention policy by keeping only the 7 most recent backups
4. Monitors disk space and sends email alerts if space is low
5. Checks for available system updates, particularly security updates
6. Performs log rotation to prevent logs from consuming too much disk space
7. Conducts a health check of the website by verifying HTTP status
8. Sends email alerts for critical issues
9. Maintains a detailed log of all maintenance operations

This comprehensive approach ensures the application remains healthy, secure, and backed up, with automated notifications when issues arise.

## 

## Server Testing and Validation

### Verifying Nginx Configuration

Check if Nginx is properly configured and running:

sudo systemctl status nginx

Test the Nginx configuration for syntax errors:

sudo nginx -t

### Verifying SSL Configuration

Check the SSL certificate information:

openssl s\_client -connect www.samonshahzad.com:443 -servername www.samonshahzad.com

Verify the certificate is valid and properly configured.

### Testing the Flask Application

Check if the Flask application service is running:

sudo systemctl status samoonevents

Verify the application is accessible through the domain:

curl -I https://www.samonshahzad.com

This should return a 200 OK status code.

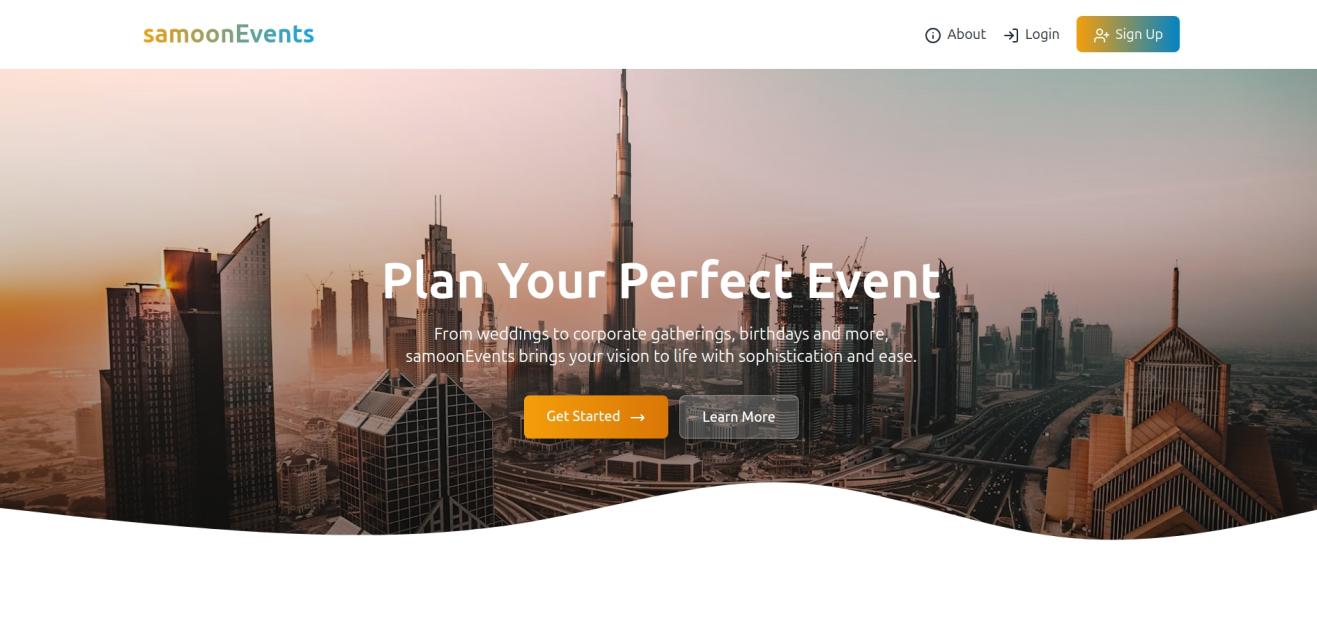
### Testing Database Connection

Verify the application can connect to the database by checking for successful queries in the logs:

sudo journalctl -u samoonevents | grep -i "database"

### Screenshots of Working Application

Homepage Screenshot



## Maintenance Documentation

### Regular Updates

To keep the server secure and up-to-date, run the following commands periodically:

sudo apt update  
sudo apt upgrade -y  
sudo apt autoremove -y

### Application Updates

To update the samoonEvents application:

cd ~/samoonEvents  
source .venv/bin/activate  
  
# Backup the database first  
cp samoonEvents/app.db samoonEvents/app.db.backup  
  
# Pull latest code and update  
git pull  
pip install -r requirements.txt  
  
# Apply any database migrations  
export FLASK\_APP=samoonEvents.app  
flask db migrate -m "Update migration"  
flask db upgrade  
  
# Restart the service  
sudo systemctl restart samoonevents

### Database Maintenance

For SQLite database maintenance:

cd ~/samoonEvents  
source .venv/bin/activate  
sqlite3 samoonEvents/app.db "VACUUM;"

This command optimizes the database by rebuilding it to reclaim unused space.

### Nginx and SSL Maintenance

Nginx configuration changes:

sudo vim /etc/nginx/sites-available/samoonevents  
sudo nginx -t  
sudo systemctl restart nginx

SSL certificate renewal (happens automatically with cron, but can be manually triggered):

sudo certbot renew

## Cost Analysis

The following table outlines the estimated monthly costs for maintaining this server:

| Service | Configuration | Monthly Cost (USD) |
| --- | --- | --- |
| AWS EC2 | t2.micro | $8.50 (on-demand) or Free Tier eligible |
| Domain Name | samonshahzad.com | ~1.25/month) |
| SSL Certificate | Let’s Encrypt | Free |
| Data Transfer | 100 GB/month | ~$9.00 |
| **Total Estimated Cost** |  | **~$18.75/month** |

Note: Costs may vary based on actual usage and AWS pricing changes. Free tier eligibility can significantly reduce first-year costs.

### Cost Optimization Strategies

1. **Use Reserved Instances**: For long-term deployments, reserved instances can reduce EC2 costs by up to 75%
2. **Scale appropriately**: The t2.micro instance is sufficient for development and low traffic, but may need to be upgraded for production
3. **Monitor data transfer**: Set up AWS CloudWatch alarms to monitor and alert on excessive data transfer usage
4. **Use AWS Free Tier**: Take advantage of AWS Free Tier offerings for the first 12 months

## Troubleshooting Common Issues

### Nginx 502 Bad Gateway Error

This typically means Gunicorn isn’t running or accessible.

Check the Flask application service:

sudo systemctl status samoonevents

If the service is failing, check the logs:

sudo journalctl -u samoonevents

Common solutions:

1. Check that the wsgi.py file is correctly configured
2. Ensure the virtual environment paths in the service file are correct
3. Verify that all dependencies are installed:

* source .venv/bin/activate  
  pip install -r requirements.txt

### Database Connection Issues

If the application cannot connect to the database:

1. Check the database file exists:

* ls -la ~/samoonEvents/samoonEvents/app.db

1. Verify the permissions:

* sudo chown -R ubuntu:ubuntu ~/samoonEvents

1. Check for database lock issues:

* find ~/ -name "app.db-\*"
* If lock files exist, you may need to remove them and restart the application.

### SSL Certificate Issues

If your certificate isn’t renewing or shows as invalid:

Check Certbot certificates:

sudo certbot certificates

Try manual renewal:

sudo certbot renew --force-renewal

Check Nginx SSL configuration:

grep -r "ssl" /etc/nginx/sites-available/

### 

### Application Loading Errors

If the application loads but has errors:

1. Check the application logs:

* sudo journalctl -u samoonevents

1. Test the application directly with Gunicorn:

* cd ~/samoonEvents  
  source .venv/bin/activate  
  gunicorn --bind 0.0.0.0:5000 wsgi:app

1. Check for static file issues:

* sudo ls -la /etc/nginx/sites-available/samoonevents  
  sudo ls -la ~/samoonEvents/samoonEvents/static

## References

1. AWS Documentation. “Amazon EC2 User Guide.” Retrieved from https://docs.aws.amazon.com/ec2/
2. Nginx Documentation. “Beginner’s Guide.” Retrieved from https://nginx.org/en/docs/beginners\_guide.html
3. Flask Documentation. “Deployment Options.” Retrieved from https://flask.palletsprojects.com/en/2.2.x/deploying/
4. SQLAlchemy Documentation. “SQLAlchemy 2.0 Documentation.” Retrieved from https://docs.sqlalchemy.org/en/20/
5. Let’s Encrypt. “Getting Started.” Retrieved from https://letsencrypt.org/getting-started/
6. DigitalOcean Community Tutorials. “How To Serve Flask Applications with Gunicorn and Nginx on Ubuntu 20.04.” Retrieved from https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-gunicorn-and-nginx-on-ubuntu-20-04
7. Certbot Documentation. “User Guide.” Retrieved from https://certbot.eff.org/docs/using.html
8. Python Documentation. “venv — Creation of virtual environments.” Retrieved from https://docs.python.org/3/library/venv.html