Roblox Studio AI Plugin - Deployment and Hosting Guide

Author: Manus AI
Date: June 6, 2025
Version: 1.0

Table of Contents

Introduction

This guide provides comprehensive instructions for deploying and hosting the Roblox Studio AI Plugin server. The server component is essential for the plugin to function, as it handles user authentication, AI model integration, and file management. This guide covers various free hosting options, AI API configuration, deployment instructions, and troubleshooting tips.

The Roblox Studio AI Plugin consists of two main components:

1. **Client-side Plugin**: A Lua script that runs within Roblox Studio and provides the user interface.
2. **Server-side API**: A Python Flask application that handles authentication, AI model integration, and file management.

This guide focuses on deploying the server-side component, which is required for the plugin to function properly. The server needs to be accessible from the internet so that the Roblox Studio plugin can communicate with it.

Server Requirements

The server component has the following requirements:

- **Python 3.7+**: The server is built using Python and requires version 3.7 or higher.
- **Flask**: A lightweight web framework used for the API endpoints.
- **OpenAI API**: Used to communicate with AI models through OpenRouter or other providers.
- **Storage**: For user data, logs, and uploaded files.
- **Memory**: At least 512MB RAM (1GB recommended).
- **CPU**: Basic CPU capabilities (1 vCPU is sufficient).
- **Network**: Public internet access with HTTPS support.

The server has been designed to be lightweight and can run on most free hosting platforms with minimal resources. The main consideration is ensuring that the server is publicly accessible and can handle the expected number of requests.

Free Hosting Options

There are several free hosting options available for deploying the Roblox Studio AI Plugin server. Each has its own advantages and limitations, which are detailed below.

Railway

Railway is a modern platform for deploying applications with a generous free tier that's well-suited for the Roblox Studio AI Plugin server.

Features: - Easy GitHub integration - Automatic HTTPS - Custom domains - Environment variables - Persistent storage (limited) - Built-in CI/CD

Free Tier Limitations: - $5 worth of resources per month - Projects sleep after inactivity - Limited compute hours - 512MB RAM / 1 vCPU - 1GB persistent storage

Ideal For: - Development and testing - Small to medium user bases - Projects that don't require 24/7 uptime

Render

Render is a unified cloud platform that offers free web services with automatic HTTPS and global CDN.

Features: - GitHub integration - Automatic HTTPS - Global CDN - Free custom domains - Environment variables - CI/CD integration

Free Tier Limitations: - 750 hours of runtime per month - Services sleep after 15 minutes of inactivity - 512MB RAM / 0.5 CPU - Build time limited to 400 minutes per month - No persistent storage (files are ephemeral)

Ideal For: - Development and testing - Projects with intermittent usage - Stateless applications

Fly.io

Fly.io allows you to deploy applications globally with a generous free tier.

Features: - Global deployment - Automatic HTTPS - Custom domains - Persistent volumes - Docker-based deployment

Free Tier Limitations: - 3 shared-cpu-1x VMs - 3GB persistent volume storage total - 160GB outbound data transfer - Limited to 2,000 launch credits per month

Ideal For: - Global distribution - Docker-based applications - Applications requiring persistent storage

Heroku

Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud.

Features: - Easy deployment - GitHub integration - Add-ons ecosystem - Custom domains - Environment variables

Free Tier Limitations: - 550-1,000 dyno hours per month - Apps sleep after 30 minutes of inactivity - No persistent storage - Limited to 10 apps per account

Ideal For: - Quick prototyping - Simple applications - Development and testing

Comparison of Free Hosting Options

Feature	Railway	Render	Fly.io	Heroku
Free Compute	$5/month	750 hours/ month	3 shared VMs	550-1,000 hours/ month
Memory	512MB	512MB	256MB	512MB
Storage	1GB	Ephemeral	3GB total	Ephemeral
Sleep After Inactivity	Yes	15 minutes	No	30 minutes
Custom Domain	Yes	Yes	Yes	Yes
Automatic HTTPS	Yes	Yes	Yes	Yes
GitHub Integration	Yes	Yes	Yes	Yes
Environment Variables	Yes	Yes	Yes	Yes
Ease of Deployment	★★★★★	★★★★☆	★★★☆☆	★★★★☆
Best For	General use	Stateless apps	Global distribution	Quick prototyping

Based on the comparison, Railway offers the best balance of features, ease of use, and resources for the Roblox Studio AI Plugin server. However, any of these options can work depending on your specific needs and constraints.

AI API Configuration

The Roblox Studio AI Plugin server supports multiple AI providers, each with its own configuration requirements. This section covers how to set up and configure each supported provider.

OpenRouter

OpenRouter is a unified API that provides access to various AI models, including those from OpenAI, Anthropic, and other providers. It's the default provider used by the plugin and offers a free tier with limited usage.

Setup Instructions:

1. **Create an Account**:
2. Go to OpenRouter and sign up for an account.
3. Verify your email address.
4. **Get API Key**:
5. Navigate to the API Keys section in your dashboard.
6. Create a new API key with appropriate permissions.
7. Copy the API key for later use.
8. **Configure the Server**:
9. Open the `config.py` file in the server directory.
10. Update the `OPENROUTER_API_KEY` value with your API key: `python "OPENROUTER_API_KEY": "your-api-key-here",`

Free Tier Limitations: - 20 requests per minute - 50 requests per day (for accounts with less than 10 credits) - 1,000 requests per day (for accounts with 10 or more credits) - Limited to models with `:free` suffix

Hugging Face

Hugging Face offers a free Inference API that provides access to thousands of open-source models. The plugin supports Hugging Face as an alternative to OpenRouter.

Setup Instructions:

1. **Create an Account**:
2. Go to Hugging Face and sign up for an account.
3. Verify your email address.
4. **Get API Key**:
5. Navigate to your profile settings.
6. Go to the "Access Tokens" section.
7. Create a new token with "read" scope.
8. Copy the token for later use.
9. **Configure the Server**:
10. Open the `config.py` file in the server directory.
11. Update the `HUGGINGFACE_API_KEY` value with your token: `python "HUGGINGFACE_API_KEY": "your-token-here",`

Free Tier Limitations: - ~300 requests per hour for registered users - 1 request per hour for unregistered users - Limited to open-source models

Ollama (Self-hosted)

Ollama is a self-hosted solution that allows you to run open-source models locally or on a server. This option provides unlimited usage but requires more setup and hardware resources.

Setup Instructions:

1. **Install Ollama**:
2. Go to Ollama and download the appropriate version for your server.
3. Follow the installation instructions for your platform.
4. **Download Models**:
5. Use the Ollama CLI to download the models you want to use: `bash ollama pull llama3 ollama pull mistral ollama pull codellama`
6. **Start Ollama Server**:
7. Run the Ollama server: `bash ollama serve`

8. **Configure the Server**:

9. Open the `config.py` file in the server directory.
10. Update the `OLLAMA_BASE_URL` value with your Ollama server URL: `python "OLLAMA_BASE_URL": "http://your-server-ip:11434",`

Requirements: - Server with decent CPU (GPU recommended for better performance) - At least 8GB RAM (16GB+ recommended) - Sufficient storage for models (5-10GB per model)

Deployment Instructions

This section provides step-by-step instructions for deploying the Roblox Studio AI Plugin server on various hosting platforms.

Railway Deployment

Railway offers one of the easiest deployment experiences and is recommended for most users.

1. **Prepare Your Repository**:
2. Create a GitHub repository with your server code.

3. Ensure your repository includes:

    - `app.py`
    - `config.py`
    - `requirements.txt`
    - `Procfile`
    - Other server files

4. **Sign Up for Railway**:

5. Go to Railway and sign up with your GitHub account.

6. Authorize Railway to access your repositories.

7. **Create a New Project**:

8. Click on "New Project" in the Railway dashboard.
9. Select "Deploy from GitHub repo".

10. Choose your repository from the list.

11. **Configure Environment Variables**:

12. Click on your newly created project.
13. Go to the "Variables" tab.
14. Add the following environment variables:
    - `PORT`: `5000`
    - `OPENROUTER_API_KEY`: Your OpenRouter API key
    - `HUGGINGFACE_API_KEY`: Your Hugging Face API key (optional)
15. **Deploy the Project**:
16. Railway will automatically deploy your project based on the repository.
17. Wait for the deployment to complete.
18. **Get Your Domain**:
19. Once deployed, go to the "Settings" tab.
20. Find the "Domains" section.
21. Railway provides a default domain (e.g., `your-app-name.up.railway.app`).
22. You can also configure a custom domain if desired.
23. **Test the Deployment**:
24. Visit your domain in a web browser.
25. You should see a response from the server.

Render Deployment

Render is another excellent option for hosting the Roblox Studio AI Plugin server.

1. **Prepare Your Repository**:
2. Create a GitHub repository with your server code.
3. Ensure your repository includes:
    - `app.py`
    - `config.py`
    - `requirements.txt`
    - Other server files
4. **Sign Up for Render**:
5. Go to Render and sign up with your GitHub account.

6. Authorize Render to access your repositories.
7. **Create a New Web Service**:
8. Click on "New" in the Render dashboard.
9. Select "Web Service".
10. Choose your repository from the list.
11. **Configure the Service**:
12. Name: Choose a name for your service.
13. Environment: Select "Python".
14. Build Command: `pip install -r requirements.txt`
15. Start Command: `gunicorn app:app`
16. Select the free plan.
17. **Configure Environment Variables**:
18. Scroll down to the "Environment" section.
19. Add the following environment variables:
    - `PORT`: `5000`
    - `OPENROUTER_API_KEY`: Your OpenRouter API key
    - `HUGGINGFACE_API_KEY`: Your Hugging Face API key (optional)
20. **Deploy the Service**:
21. Click "Create Web Service".
22. Wait for the deployment to complete.
23. **Get Your Domain**:
24. Once deployed, Render provides a default domain (e.g., `your-app-name.onrender.com`).
25. You can also configure a custom domain in the service settings.
26. **Test the Deployment**:
27. Visit your domain in a web browser.
28. You should see a response from the server.

Fly.io Deployment

Fly.io offers global deployment with a generous free tier.

1. **Install Flyctl**:
2. Install the Fly.io CLI tool by following the instructions on the Fly.io website.
3. **Authenticate**:
4. Run `flyctl auth login` to authenticate with your Fly.io account.
5. **Prepare Your Project**:
6. Create a `fly.toml` file in your project directory: ```toml app = "roblox-ai-plugin"

   [build] builder = "paketobuildpacks/builder:base"

   [env] PORT = "5000"

   [http_service] internal_port = 5000 force_https = true

   [[services]] protocol = "tcp" internal_port = 5000

   [[services.ports]] port = 80 handlers = ["http"]

   [[services.ports]] port = 443 handlers = ["tls", "http"] ```
7. **Deploy the Application**:
8. Run `flyctl launch` in your project directory.
9. Follow the prompts to configure your application.
10. When asked about a Postgres database, select "No".
11. When asked about deploying now, select "Yes".
12. **Set Environment Variables**:
13. Run the following commands to set your API keys: `bash flyctl secrets set OPENROUTER_API_KEY=your-api-key-here flyctl secrets set HUGGINGFACE_API_KEY=your-token-here`
14. **Get Your Domain**:
15. Run `flyctl info` to get information about your deployment, including the domain.
16. The domain will be in the format `your-app-name.fly.dev`.

17. **Test the Deployment**:

18. Visit your domain in a web browser.
19. You should see a response from the server.

Heroku Deployment

Heroku is a well-established platform for hosting web applications.

1. **Install Heroku CLI**:

2. Install the Heroku CLI tool by following the instructions on the Heroku website.

3. **Authenticate**:

4. Run `heroku login` to authenticate with your Heroku account.

5. **Create a New App**:

6. Run `heroku create roblox-ai-plugin` to create a new Heroku app.

7. This will also add a Heroku remote to your Git repository.

8. **Set Environment Variables**:

9. Run the following commands to set your API keys: `bash heroku config:set OPENROUTER_API_KEY=your-api-key-here heroku config:set HUGGINGFACE_API_KEY=your-token-here`

10. **Deploy the Application**:

11. Run `git push heroku main` to deploy your application to Heroku.

12. Wait for the deployment to complete.

13. **Get Your Domain**:

14. Run `heroku domains` to get the domain for your application.

15. The domain will be in the format `your-app-name.herokuapp.com`.

16. **Test the Deployment**:

17. Visit your domain in a web browser.
18. You should see a response from the server.

Self-Hosting Instructions

If you prefer to self-host the Roblox Studio AI Plugin server, this section provides instructions for setting up a VPS, deploying with Docker, and configuring Nginx.

VPS Setup

1. **Choose a VPS Provider**:
2. Popular options include DigitalOcean, Linode, Vultr, AWS Lightsail, and Google Cloud Platform.

3. For the Roblox Studio AI Plugin server, a basic plan with 1GB RAM and 1 vCPU is sufficient.

4. **Create a VPS Instance**:

5. Sign up for an account with your chosen provider.
6. Create a new VPS instance with Ubuntu 20.04 or later.

7. Set up SSH keys for secure access.

8. **Connect to Your VPS**:

9. Use SSH to connect to your VPS: `bash ssh root@your-server-ip`

10. **Update the System**:

11. Update the package list and upgrade installed packages: `bash apt update && apt upgrade -y`

12. **Install Required Packages**:

13. Install Python, pip, and other required packages: `bash apt install -y python3 python3-pip python3-venv nginx certbot python3-certbot-nginx`

Docker Deployment

Using Docker is the recommended approach for self-hosting the Roblox Studio AI Plugin server.

1. **Install Docker**:
2. Install Docker on your VPS: `bash apt install -y docker.io docker-compose`

3. Start and enable Docker: `bash systemctl start docker systemctl enable docker`
4. **Create a Docker Compose File**:
5. Create a directory for your project: `bash mkdir -p /opt/roblox-ai-plugin cd /opt/roblox-ai-plugin`
6. Create a `docker-compose.yml` file: ```yaml version: '3'

   services: app: build: . ports: - "5000:5000" environment: - PORT=5000 - OPENROUTER_API_KEY=your-api-key-here - HUGGINGFACE_API_KEY=your-token-here volumes: - ./data:/app/data restart: always ```
7. **Create a Dockerfile**:
8. Create a `Dockerfile` in the same directory: ```dockerfile FROM python:3.9-slim

   WORKDIR /app

   COPY requirements.txt . RUN pip install --no-cache-dir -r requirements.txt

   COPY . .

   CMD ["gunicorn", "--bind", "0.0.0.0:5000", "app:app"] ```
9. **Copy Your Server Files**:
10. Copy your server files to the VPS: `bash # On your local machine scp -r server/* root@your-server-ip:/opt/roblox-ai-plugin/`
11. **Build and Start the Docker Container**:
12. Build and start the Docker container: `bash cd /opt/roblox-ai-plugin docker-compose up -d`
13. **Test the Deployment**:
14. Check if the container is running: `bash docker-compose ps`
15. Test the API: `bash curl http://localhost:5000/`

Nginx Configuration

Nginx is used as a reverse proxy to forward requests to your application and handle SSL termination.

1. **Create an Nginx Configuration File**:
2. Create a new Nginx configuration file: `bash nano /etc/nginx/sites-available/roblox-ai-plugin`
3. Add the following configuration: ```nginx server { listen 80; server_name your-domain.com;

```
  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;
  }
```

}```

4. **Enable the Configuration**:
5. Create a symbolic link to enable the configuration: `bash ln -s /etc/nginx/sites-available/roblox-ai-plugin /etc/nginx/sites-enabled/`
6. Test the Nginx configuration: `bash nginx -t`
7. Reload Nginx: `bash systemctl reload nginx`
8. **Set Up SSL with Certbot**:
9. Obtain an SSL certificate using Certbot: `bash certbot --nginx -d your-domain.com`
10. Follow the prompts to complete the SSL setup.
11. **Test the Deployment**:
12. Visit your domain in a web browser.
13. You should see a response from the server over HTTPS.

Plugin Configuration

After deploying the server, you need to configure the Roblox Studio plugin to connect to your server.

1. **Open the Plugin Code**:
2. Open the `plugin.lua` file in a text editor.
3. **Update the Server URL**:
4. Find the line that defines the server URL: `lua local SERVER_URL = "https://web-production-4471.up.railway.app"`
5. Replace it with your server URL: `lua local SERVER_URL = "https://your-domain.com"`
6. **Save the Plugin**:
7. Save the file and export it as an RBXMX file.
8. Install the plugin in Roblox Studio.
9. **Test the Plugin**:
10. Open Roblox Studio and click on the plugin button.
11. The plugin should connect to your server and authenticate.

Security Considerations

When deploying the Roblox Studio AI Plugin server, consider the following security best practices:

1. **API Keys**:
2. Store API keys as environment variables, not in the code.
3. Use different API keys for development and production.
4. Regularly rotate API keys.
5. **HTTPS**:
6. Always use HTTPS for production deployments.
7. Obtain and renew SSL certificates using Certbot or similar tools.
8. **Authentication**:

9. Implement rate limiting to prevent abuse.
10. Use secure authentication mechanisms.

11. Regularly review and update the authorized users list.

12. **Data Storage**:

13. Store sensitive data securely.
14. Implement proper backup procedures.

15. Consider encryption for sensitive data.

16. **Updates**:

17. Regularly update dependencies to patch security vulnerabilities.
18. Monitor security advisories for used packages.

Troubleshooting

This section covers common issues and their solutions when deploying the Roblox Studio AI Plugin server.

Connection Issues

Problem: The plugin cannot connect to the server.

Solutions: - Verify that the server is running and accessible from the internet. - Check that the server URL in the plugin code is correct. - Ensure that the server is using HTTPS if required. - Check for firewall or network restrictions.

Authentication Issues

Problem: Users cannot authenticate with the server.

Solutions: - Verify that the user is in the authorized users list. - Check the server logs for authentication errors. - Ensure that the user's username is spelled correctly.

AI Model Issues

Problem: AI model responses are not working.

Solutions: - Verify that the API keys are correct and have sufficient credits. - Check the server logs for API errors. - Try a different AI model or provider. - Check for rate limiting issues.

Deployment Issues

Problem: The server fails to deploy.

Solutions: - Check the deployment logs for errors. - Verify that all required files are included in the repository. - Ensure that the `requirements.txt` file includes all dependencies. - Check for compatibility issues with the hosting platform.

Maintenance and Updates

To keep your Roblox Studio AI Plugin server running smoothly, follow these maintenance best practices:

1. **Regular Updates**:
2. Regularly update dependencies to patch security vulnerabilities.
3. Monitor for updates to the plugin and server code.
4. Test updates in a development environment before deploying to production.
5. **Monitoring**:
6. Set up monitoring to track server health and performance.
7. Monitor API usage to avoid exceeding free tier limits.
8. Set up alerts for server downtime or errors.
9. **Backups**:
10. Regularly backup user data and configurations.
11. Store backups securely and test restoration procedures.
12. **Scaling**:
13. Monitor resource usage and scale as needed.
14. Consider upgrading to paid tiers if free tier limits are consistently reached.

References

1. Railway Documentation: https://docs.railway.app/
2. Render Documentation: https://render.com/docs
3. Fly.io Documentation: https://fly.io/docs/
4. Heroku Documentation: https://devcenter.heroku.com/
5. OpenRouter Documentation: https://openrouter.ai/docs

6. Hugging Face Documentation: https://huggingface.co/docs
7. Ollama Documentation: https://ollama.ai/docs
8. Flask Documentation: https://flask.palletsprojects.com/
9. Nginx Documentation: https://nginx.org/en/docs/
10. Docker Documentation: https://docs.docker.com/