Bot Setup Guide

Prerequisites

Before proceeding, ensure you have the following:

- A Discord Developer account
- A Twitch Developer account
- OpenAI API key
- A server or cloud service for deployment (e.g., Render, AWS, Heroku, or a self-hosted server)
- Node.js or Python (depending on your bot's language)

---

1. Setting Up Environment Variables

Create a `.env` file in the root directory of your project and add the following:

```
DISCORD_TOKEN=your_discord_bot_token
DISCORD_CLIENT_ID=your_discord_client_id
DISCORD_GUILD_ID=your_discord_server_id
TWITCH_CLIENT_ID=your_twitch_client_id
TWITCH_CLIENT_SECRET=your_twitch_client_secret
TWITCH_OAUTH_TOKEN=your_twitch_oauth_token
OPENAI_API_KEY=your_openai_api_key
WEBHOOK_URL=your_webhook_url (if applicable)
```

Ensure this file is included in your `.gitignore` to prevent accidental leaks.

---

2. File Structure

Your project should have a structure similar to this:

```
/root
  ├── src/
  │     ├── bot.js (or bot.py)
  │     ├── commands/
  │     ├── events/
  │     ├── integrations/
  │     ├── utils/
  │
  ├── config/
  │     ├── config.json
  │     ├── settings.json
  │
  ├── .env
  ├── package.json (for Node.js)
  ├── requirements.txt (for Python)
  ├── README.md
  ├── .gitignore
```

Modify this structure based on your bot's needs.

---

3. Discord Developer Portal Setup

1. Go to [Discord Developer Portal]
2. Click **New Application**
3. Name your bot and save
4. Navigate to **Bot** -> **Add Bot**
5. Copy the **Token** and add it to your `.env` file
6. Enable necessary intents under **Privileged Gateway Intents**
7. Navigate to **OAuth2** -> **URL Generator**
   - Select **bot** and **applications.commands**
   - Choose necessary permissions
   - Copy and use the generated invite link to add the bot to your server

---

4. Twitch Developer Setup

1. Go to Twitch Developer Console
2. Create a new application
3. Set the **OAuth Redirect URL** to match your bot's web service URL (or `http://localhost` for local testing)
4. Copy **Client ID** and **Client Secret** to your `.env` file
5. Obtain an OAuth token:

Use a Twitch authentication service or generate via API call:
 curl -X POST "https://id.twitch.tv/oauth2/token" \
-d "client_id=your_client_id" \
-d "client_secret=your_client_secret" \
-d "grant_type=client_credentials"

-
- Copy the returned `access_token` and update your `.env`

---

5. Deploying Your Bot to a Web Service

Option 1: Deploying on Render (Recommended)

1. Push your code to GitHub
2. Go to Render
3. Create a new **Web Service**
4. Connect your GitHub repository
5. Set environment variables in the Render dashboard
6. Choose a start command (e.g., `node src/bot.js` or `python src/bot.py`)
7. Deploy and monitor logs for errors

Option 2: Deploying on a VPS or Self-Hosted Server

1. SSH into your server
2. Clone your repository

Install dependencies:
 npm install # For Node.js
pip install -r requirements.txt # For Python

3.
4. Set environment variables using `export` or a `.env` file

Start the bot using `pm2` (for Node.js) or `screen/tmux`:
 pm2 start src/bot.js --name bot # Node.js
python3 src/bot.py & # Python

5.

---

6. Testing and Troubleshooting

- Use `console.log()` or `print()` statements to debug issues
- Check the logs in your web service or `pm2 logs bot`
- If the bot doesn't respond, ensure:
    - Tokens and API keys are correctly set
    - Required permissions are granted
    - The bot is online and running

Conclusion

By following this guide, you should have a fully operational bot integrated with Discord, Twitch, and ChatGPT. If you run into any issues, check the respective developer portals for additional troubleshooting.

Happy coding!