How to Deploy Your Discord Bot on AWS EC2 for 24/7 Availability

Discord bots are a powerful way to automate tasks, engage users, and add functionality to your Discord server. If you've built a bot and want it to run

continuously without relying on your local machine, deploying it on AWS EC2 is a robust and scalable solution.

In this tutorial, you'll learn how to deploy your Discord bot on AWS EC2, ensuring it's always available to serve your community. By the end, you'll have the confidence to:

- Set up an AWS EC2 instance for hosting your bot.
- Configure it to run your bot script continuously.
- Restart it automatically if something goes wrong.

I've written this guide because I've faced these challenges myself and know how daunting cloud deployment can seem for first-timers. With years of experience in both bot development and cloud platforms, I'll walk you through this process step by step.

Prerequisites

Before diving in, ensure you have:

- A functional Discord bot script. Please check out https://www.startanalytics.net/blog if you want to make the bot script first
- An AWS account.
- Basic familiarity with the command line and Python.

Step 1: Setting Up Your AWS EC2 Instance

1. **Log into AWS**: Go to AWS Console and sign in.
2. Select your server location, you will want to pick a location close to you
3. **Launch an EC2 Instance**:
    - Navigate to the EC2 dashboard and click "Launch Instances."
    - Choose an Amazon Machine Image (AMI). For most bots, the **Ubuntu Server 22.04 LTS** works well.
    - Select an instance type. The **t2.micro** (free tier eligible) is sufficient for small bots.

- Configure instance details, leaving most settings as default. Ensure your bot's language runtime (e.g., Python) is supported.
- Add a storage volume (default is fine for most bots).
- Add a key pair for SSH access. Download the private key file (e.g., `key.pem`).
- Configure security group rules:
  - Allow **SSH** (port 22) from your IP.
  - Allow your bot's communication port (e.g., 443 for HTTPS or 80 for HTTP).

4. **Review and Launch**: Launch your instance and note its public IP address.

Step 2: Connecting to Your Instance

1. **Open your terminal** (or an SSH client like PuTTY).
2. Run the following command, replacing `key.pem` and `public-ip` with your values:

```
ssh -i "key.pem" ubuntu@your-public-ip
```

3. Once connected, update the system:

```
sudo apt update && sudo apt upgrade -y
```

Step 3: Installing Dependencies

1. Install the language runtime for your bot (e.g., Python):

```
sudo apt install python3 python3-pip -y
```

2. Install Git if your bot's code is in a repository:

```
sudo apt install git -y
```

3. Clone your bot's repository:

```
git clone https://github.com/your-repo.git
```

Step 4: Running Your Bot

1. Navigate to your bot's directory:

```
cd your-repo
```

2. Install your bot's dependencies:

```
pip install -r requirements.txt
```

3. Run your bot:

```
python3 bot.py
```

4. Test if your bot responds in Discord.

Step 5: Keeping Your Bot Running

To ensure your bot stays online, use **PM2** (a process manager):

1. Install PM2:

```
sudo npm install pm2@latest -g
```

2. Start your bot with PM2:

```
pm2 start bot.py --interpreter python3
```

3. Save the PM2 process list and enable it to start on reboot:

```
pm2 save
```

```
pm2 startup
```

Step 6: Securing Your Setup

1. **Use a Virtual Environment**: Avoid running your bot globally by isolating dependencies:

   ```
   python3 -m venv venv
   source venv/bin/activate
   pip install -r requirements.txt
   ```

2. **Restrict SSH Access**: Modify the security group to allow SSH only from your trusted IP addresses.
3. **Monitor Logs**: Use PM2 to view logs and debug issues:

   ```
   pm2 logs
   ```

Step 7: Automating Restarts

PM2 handles unexpected crashes. For additional resilience, schedule periodic restarts:

```
pm2 restart bot.py --cron "0 0 * * *"
```

This example restarts the bot daily at midnight.

Conclusion

By following this guide, you've successfully deployed your Discord bot on AWS EC2, ensuring it's always available to your server. This setup scales with your needs and is robust enough to handle most use cases.

Next Steps:

- Optimize your bot's performance for larger-scale deployments.

- Explore AWS monitoring tools like CloudWatch for deeper insights.
- Secure your instance further using IAM roles and encrypted environment variables.

Happy botting!

References

1. AWS EC2 Documentation
2. Discord Developer Portal
3. PM2 Documentation