Google Cloud Platform (GCP) Deployment Guide

A step-by-step guide to deploy your Discord Voice Recording Bot on Google Cloud Platform using the Always Free tier.

📋 Table of Contents

Prerequisites

Before you begin, make sure you have:

- ☑ A Google account
- ☑ Credit card (required for account verification, but won't be charged if you stay within free tier)
- ☑ Discord Bot Token and Client ID
- ☑ Google Gemini API Key
- ☑ (Optional) Google Drive folder for uploads (if using Drive upload feature)

Step 1: Create GCP Account

1. Go to Google Cloud Platform
2. Click **"Get started for free"** or sign in with your Google account
3. Complete the account setup:
    - Enter your country and accept terms
    - Add billing information (credit card required for verification)
    - **Don't worry** - you won't be charged as long as you stay within free tier limits
4. You'll receive $300 in free credits (valid for 90 days), but we'll use the Always Free tier which doesn't expire

---

Step 2: Create a Project

1. In the GCP Console, click the **project dropdown** at the top (next to "Google Cloud")
2. Click **"New Project"**
3. Enter a project name (e.g., `discord-bot`)
4. Click **"Create"**
5. Wait a few seconds, then select your new project from the dropdown

---

Step 3: Enable Compute Engine API

1. In the left sidebar, click **"APIs & Services"** → **"Library"**
2. Search for **"Compute Engine API"**
3. Click on **"Compute Engine API"**
4. Click **"Enable"**
5. Wait for the API to enable (takes about 30 seconds)

Step 4: Create Virtual Machine

4.1 Navigate to Compute Engine

1. In the left sidebar, click **"Compute Engine"** → **"VM instances"**
2. If prompted, click **"Enable"** to activate Compute Engine

4.2 Create Instance

Click the **"Create Instance"** button at the top.

4.3 Configure Instance Settings

Basic Settings:

- **Name**: `discord-bot` (or any name you prefer)
- **Region**: Choose one of these free tier regions:
  - `us-west1` (Oregon) - Recommended
  - `us-central1` (Iowa)
  - `us-east1` (South Carolina)
- **Zone**: Any zone in your selected region (e.g., `us-west1-a`)

Machine Configuration:

- **Machine family**: General-purpose
- **Series**: **E2**
- **Machine type**: **e2-micro**
  - This shows: "1 vCPU, 1 GB memory" - This is the free tier option
  - Cost should show as "Free tier eligible"

Boot Disk:

1. Click **"Change"** under Boot disk
2. Select:
   - **Operating System**: Ubuntu

- **Version**: Ubuntu 22.04 LTS
- **Boot disk type**: Standard persistent disk
- **Size**: 20 GB (free tier includes 30 GB total)

3. Click **"Select"**

Firewall:

- ☑ Check **"Allow HTTP traffic"** (optional, not needed for Discord bot)
- ☑ Check **"Allow HTTPS traffic"** (optional, not needed for Discord bot)

Note: Your bot only needs outbound internet access. No inbound ports are required.

4.4 Create the Instance

1. Scroll down and click **"Create"**
2. Wait 1-2 minutes for the VM to be created
3. You'll see a green checkmark when it's ready

4.5 Note Your External IP

1. In the VM instances list, find your instance
2. Copy the **External IP** address (you'll need this for SSH)

Step 5: Connect to Your VM

Option A: Browser SSH (Easiest)

1. In the VM instances list, click the **"SSH"** button next to your instance
2. A browser terminal window will open
3. You're now connected! ☑

Option B: SSH from Terminal (Advanced)

If you prefer using your local terminal:

1. Click the dropdown arrow next to "SSH" → **"View gcloud command"**
2. Copy the command shown (looks like: `gcloud compute ssh discord-bot --zone=us-west1-a`)
3. Run it in your local terminal (requires gcloud CLI)

Step 6: Install Dependencies

Once connected via SSH, run these commands one by one:

6.1 Update System Packages

```
sudo apt-get update
sudo apt-get upgrade -y
```

This may take a few minutes.

6.2 Install Node.js 18.x

```
# Add NodeSource repository
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -

# Install Node.js
sudo apt-get install -y nodejs

# Verify installation
node --version
npm --version
```

You should see `v18.x.x` for Node.js.

6.3 Install Python and pip

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

# Verify installation
python3 --version
pip3 --version
```

You should see Python 3.10+.

6.4 Install FFmpeg

```
sudo apt-get install -y ffmpeg

# Verify installation
ffmpeg -version
```

You should see FFmpeg version information.

6.5 Install Git

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

6.6 Install Build Tools

```
sudo apt-get install -y build-essential
```

These are needed for compiling native Node.js modules.

Step 7: Deploy Your Bot

7.1 Create Bot Directory

```
# Create directory for your bot
mkdir -p ~/bots
cd ~/bots
```

7.2 Clone Your Repository

Option A: Using Git (Recommended)

```
# Replace with your actual repository URL
git clone https://github.com/your-username/your-repo.git Discord_Bot
cd Discord_Bot
```

Option B: Upload Files Manually

If your code isn't in a repository:

1. On your local machine, compress your bot files into a zip file
2. In GCP Console, go to **Compute Engine** → **VM instances**
3. Click your instance → **"Edit"** → Scroll to **"SSH Keys"**
4. Or use `scp` from your local machine:

   ```
   scp -r /path/to/your/bot/files username@EXTERNAL_IP:~/bots/Discord_Bot
   ```

7.3 Install Node.js Dependencies

```
cd ~/bots/Discord_Bot
npm install
```

This will install all packages from `package.json`. Wait for it to complete.

7.4 Install Python Dependencies

```
pip3 install -r requirements.txt
```

7.5 Create Environment File

```
nano .env
```

Paste this template and fill in your values:

```
# Discord Configuration
DISCORD_TOKEN=your_discord_bot_token_here
DISCORD_CLIENT_ID=your_discord_client_id_here
DISCORD_GUILD_ID=your_guild_id_here

# Google Gemini API
GEMINI_API_KEY=your_gemini_api_key_here

# Google Drive (Optional - only if using uploads)
# Get this from your Google Drive folder URL after sharing with service account
FOLDER_ID=your_google_drive_folder_id_here

# Service Account Key Path (Optional - defaults to service-account-key.json in bot
directory)
# Only needed if you want to use a different path or filename
# SERVICE_ACCOUNT_KEY_PATH=/path/to/service-account-key.json

# Data Directory (Optional - defaults to current directory)
DATA_DIR=/home/your-username/bots/Discord_Bot
```

To save and exit nano:

- Press `Ctrl + X`
- Press `Y` to confirm
- Press `Enter` to save

7.6 Secure Environment File

```
chmod 600 .env
```

This makes the file readable only by you.

7.7 Create Required Directories

```
mkdir -p PCM_Files Summary
```

7.8 (Optional) Set Up Google Drive Service Account

If you want to automatically upload summaries to Google Drive, set up a Service Account (recommended for GCP hosting):

Step 1: Enable Google Drive API

1. In GCP Console, go to **"APIs & Services"** → **"Library"**
2. Search for **"Google Drive API"**
3. Click on it and click **"Enable"**

Step 2: Create Service Account

1. In GCP Console, go to **"IAM & Admin"** → **"Service Accounts"**
2. Click **"Create Service Account"**
3. Fill in the details:
   - **Service account name**: `discord-bot-drive` (or any name)
   - **Service account ID**: Auto-generated (or customize)
   - **Description**: "Service account for Discord bot Google Drive uploads"
4. Click **"Create and Continue"**

Step 3: Grant Permissions (Optional)

For basic Drive access, you can skip this step. The service account will access only folders you explicitly share with it.

If you want to grant project-level permissions:

- **Role**: Select **"Editor"** (or a more restrictive role)
- Click **"Continue"** → **"Done"**

Step 4: Create and Download Service Account Key

1. In the **Service Accounts** list, click on your newly created service account
2. Go to the **"Keys"** tab
3. Click **"Add Key"** → **"Create new key"**
4. Select **"JSON"** as the key type
5. Click **"Create"**
6. The JSON key file will automatically download to your computer

Important: Keep this file secure! It contains sensitive credentials.

Step 5: Upload Service Account Key to Your VM

Option A: Using Browser SSH (Easiest)

1. In GCP Console, click **"SSH"** to connect to your VM
2. In the browser terminal, run:

```
nano ~/bots/Discord_Bot/service-account-key.json
```

3. Open the downloaded JSON file on your local computer
4. Copy the entire contents

5. Paste into the nano editor in the browser
6. Save and exit: `Ctrl + X`, then `Y`, then `Enter`

Option B: Using SCP (From Local Terminal)

```
scp /path/to/downloaded/service-account-key.json
username@EXTERNAL_IP:~/bots/Discord_Bot/service-account-key.json
```

Step 6: Secure the Service Account Key

```
chmod 600 ~/bots/Discord_Bot/service-account-key.json
```

Step 7: Share Google Drive Folder with Service Account

1. Open Google Drive in your browser
2. Create a new folder (or use an existing one) for bot uploads
3. Right-click the folder → **"Share"**
4. In the **"Share with people and groups"** field, enter the service account email
   - The email looks like: `discord-bot-drive@your-project-id.iam.gserviceaccount.com`
   - You can find it in the JSON key file under `"client_email"`
5. Set permission to **"Editor"**
6. Uncheck **"Notify people"** (service accounts don't have email)
7. Click **"Share"**

Step 8: Get Folder ID

1. Open the shared folder in Google Drive
2. Look at the URL: `https://drive.google.com/drive/folders/FOLDER_ID_HERE`
3. Copy the `FOLDER_ID_HERE` part
4. Add it to your `.env` file:

   ```
   nano .env
   ```

   Add or update:

   ```
   FOLDER_ID=your_folder_id_here
   ```

Step 9: Test Google Drive Upload

```
cd ~/bots/Discord_Bot
node uploader.js
```

You should see:

- ☑ "Using Service Account authentication..."
- ☑ "Service Account authenticated successfully"
- ☑ Upload messages (if there are files to upload)

Note: The uploader will automatically use the service account if `service-account-key.json` is present. No OAuth flow needed!

7.9 Test Your Bot

Before setting up auto-start, test that everything works:

```
node bot.js
```

You should see:

- ☑ Bot logged in message
- ☑ Slash commands deployed
- ☑ No errors

Press `Ctrl + C` to stop the bot.

Step 8: Set Up Auto-Start

Choose one method to keep your bot running 24/7:

Option A: systemd (Recommended)

1. **Update the service file:**

```
nano discord-bot.service
```

Update these lines with your actual username and paths:

```
[Unit]
Description=Discord Voice Recording Bot
After=network.target

[Service]
Type=simple
User=YOUR_USERNAME_HERE
```

```
WorkingDirectory=/home/YOUR_USERNAME_HERE/bots/Discord_Bot
Environment="NODE_ENV=production"
EnvironmentFile=/home/YOUR_USERNAME_HERE/bots/Discord_Bot/.env
ExecStart=/usr/bin/node /home/YOUR_USERNAME_HERE/bots/Discord_Bot/bot.js
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
SyslogIdentifier=discord-bot

# Resource limits
LimitNOFILE=65536
MemoryMax=2G

[Install]
WantedBy=multi-user.target
```

To find your username:

```
whoami
```

2. **Copy service file to systemd:**

```
sudo cp discord-bot.service /etc/systemd/system/
```

3. **Reload systemd and enable service:**

```
sudo systemctl daemon-reload
sudo systemctl enable discord-bot
sudo systemctl start discord-bot
```

4. **Check status:**

```
sudo systemctl status discord-bot
```

You should see "active (running)" in green.

5. **View logs:**

```
sudo journalctl -u discord-bot -f
```

Press `Ctrl + C` to exit log view.

Option B: PM2

1. **Install PM2 globally:**

```
sudo npm install -g pm2
```

2. **Start the bot:**

```
cd ~/bots/Discord_Bot
pm2 start ecosystem.config.js
```

3. **Save PM2 configuration:**

```
pm2 save
```

4. **Set up PM2 to start on boot:**

```
pm2 startup
```

This will show a command. Copy and run it (it will look like `sudo env PATH=...`).

5. **View logs:**

```
pm2 logs discord-bot
```

Step 9: Verify Deployment

9.1 Check Bot Status

If using systemd:

```
sudo systemctl status discord-bot
```

If using PM2:

```
pm2 status
```

9.2 Check Bot in Discord

1. Open your Discord server
2. Check if the bot appears **online** (green dot)
3. Try the `/join` command in a voice channel
4. Verify the bot joins and responds

9.3 Check Logs

If using systemd:

```
sudo journalctl -u discord-bot -n 50
```

If using PM2:

```
pm2 logs discord-bot --lines 50
```

Look for:

- ☑ "Bot logged in as [BotName]"
- ☑ "Slash commands auto-deployed successfully!"
- ❌ No error messages

Step 10: Monitor & Maintain

10.1 Monitor Disk Space

With 30 GB total storage, monitor usage regularly:

```
# Check overall disk usage
df -h

# Check bot directory size
du -sh ~/bots/Discord_Bot/*
```

10.2 Set Up Automatic Cleanup

Your bot includes cleanup scripts. Set up a daily cleanup:

```
crontab -e
```

Add this line to run cleanup daily at 2 AM:

```
0 2 * * * cd /home/$(whoami)/bots/Discord_Bot && /usr/bin/node cleanup.js
```

Save and exit (Ctrl+X, Y, Enter).

10.3 Monitor Free Tier Usage

1. In GCP Console, go to **"Billing"** → **"Budgets & alerts"**
2. Create a budget alert to notify you if you approach free tier limits
3. Set alert threshold to $1 (to catch any unexpected charges)

10.4 Update Your Bot

When you need to update your bot:

```
cd ~/bots/Discord_Bot

# If using Git
git pull

# Reinstall dependencies if package.json changed
npm install

# Restart the bot
sudo systemctl restart discord-bot
# OR
pm2 restart discord-bot
```

Troubleshooting

Bot Won't Start

Check logs:

```
sudo journalctl -u discord-bot -n 100
# OR
pm2 logs discord-bot --lines 100
```

Common issues:

1. **Missing environment variables:**

   ```
   cat .env
   ```

   Make sure all required variables are set.

2. **Wrong file paths in service file:**
   - Verify username in `discord-bot.service`
   - Check that paths match your actual directory structure
3. **Permission issues:**

```
ls -la ~/bots/Discord_Bot
```

Make sure files are readable.

Bot Disconnects Frequently

1. **Check memory usage:**

```
free -h
top
```

The e2-micro has 1 GB RAM. If memory is high, consider:

   - Running cleanup more frequently
   - Reducing audio file retention
2. **Check network connectivity:**

```
ping google.com
curl https://discord.com/api/v10
```

Out of Disk Space

1. **Check disk usage:**

```
df -h
du -sh ~/bots/Discord_Bot/PCM_Files
```

2. **Run cleanup:**

```
node cleanup.js
```

3. **Manually delete old files:**

```
# Be careful! Only delete files you don't need
rm ~/bots/Discord_Bot/PCM_Files/*.pcm
```

Bot Not Responding in Discord

1. **Check if bot is online:**
   - Look for green dot in Discord
   - Check bot status: `sudo systemctl status discord-bot`
2. **Verify Discord token:**

   ```
   grep DISCORD_TOKEN .env
   ```

   Make sure the token is correct.
3. **Check Discord API status:**
   - Visit Discord Status

FFmpeg Not Found

```
which ffmpeg
ffmpeg -version
```

If not found, reinstall:

```
sudo apt-get install -y ffmpeg
```

Python/Node.js Not Found

Node.js:

```
which node
node --version
```

If not found, reinstall:

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Python:

```
which python3
python3 --version
```

If not found, reinstall:

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

Free Tier Limits

What's Included (Always Free)

- ☑ **1 e2-micro VM instance** per month (always-on)
- ☑ **30 GB-months** of standard persistent disk storage
- ☑ **1 GB** of network egress per month (from North America)
- ☑ **External IP address** (free)

Important Notes

- **Region restriction**: Must use US regions (`us-west1`, `us-central1`, `us-east1`)
- **Storage limit**: 30 GB total (boot disk + any additional disks)
- **Network limit**: 1 GB outbound data per month (Discord bot uses minimal data)
- **No time limit**: Always Free tier doesn't expire (unlike AWS free tier)

Staying Within Limits

- Monitor your usage in **Billing** → **Reports**
- Set up billing alerts
- Use cleanup scripts to manage disk space
- Your bot should easily stay within these limits

What Happens If You Exceed Limits?

- You'll be charged for usage beyond free tier
- Set up billing alerts to avoid surprises
- GCP will notify you before charges occur

Quick Reference Commands

Service Management (systemd)

```
# Start bot
sudo systemctl start discord-bot
```

```
# Stop bot
sudo systemctl stop discord-bot

# Restart bot
sudo systemctl restart discord-bot

# Check status
sudo systemctl status discord-bot

# View logs
sudo journalctl -u discord-bot -f

# View last 100 log lines
sudo journalctl -u discord-bot -n 100
```

Service Management (PM2)

```
# Start bot
pm2 start ecosystem.config.js

# Stop bot
pm2 stop discord-bot

# Restart bot
pm2 restart discord-bot

# Check status
pm2 status

# View logs
pm2 logs discord-bot

# View last 50 lines
pm2 logs discord-bot --lines 50
```

Useful System Commands

```
# Check disk space
df -h

# Check memory usage
free -h

# Check running processes
top

# Check network connectivity
ping google.com
```

```
# Find your username
whoami

# Check current directory
pwd
```

Security Best Practices

1. **Keep `.env` file secure:**

   ```
   chmod 600 .env
   ```

2. **Don't commit sensitive files:**

   - Add `.env` to `.gitignore`
   - Don't commit `service-account-key.json` (contains sensitive credentials)
   - Don't commit `token.json` or `credentials.json` (if using OAuth2 fallback)

3. **Update system regularly:**

   ```
   sudo apt-get update && sudo apt-get upgrade
   ```

4. **Use SSH keys** (already set up by default in GCP)

5. **Keep your Discord token secret** - never share it

Next Steps

- ☑ Your bot is now running 24/7 on GCP!
- 📊 Set up monitoring and alerts
- 🔄 Configure automatic backups of summaries
- 📁 Set up Google Drive uploads using Service Account (see Step 7.8) for automatic long-term storage
- 🔧 Set up log rotation if logs get large

Getting Help

If you encounter issues:

1. Check the Troubleshooting section above
2. Review bot logs for error messages
3. Check Discord.js documentation
4. Verify all environment variables are set correctly
5. Check GCP documentation

Summary

You've successfully deployed your Discord bot on Google Cloud Platform! 🎉

What you've accomplished:

- ☑ Created a GCP account and project
- ☑ Set up a free e2-micro VM instance
- ☑ Installed all required dependencies
- ☑ Deployed your bot
- ☑ Configured auto-start with systemd or PM2
- ☑ Verified the bot is running

Your bot is now running 24/7 on Google's infrastructure, completely free (as long as you stay within free tier limits)!

Last Updated: 2024 **GCP Free Tier:** Always Free (no expiration)