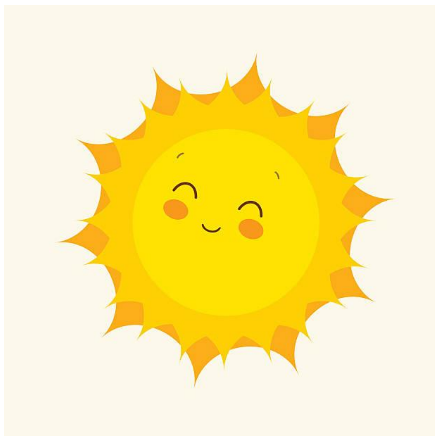
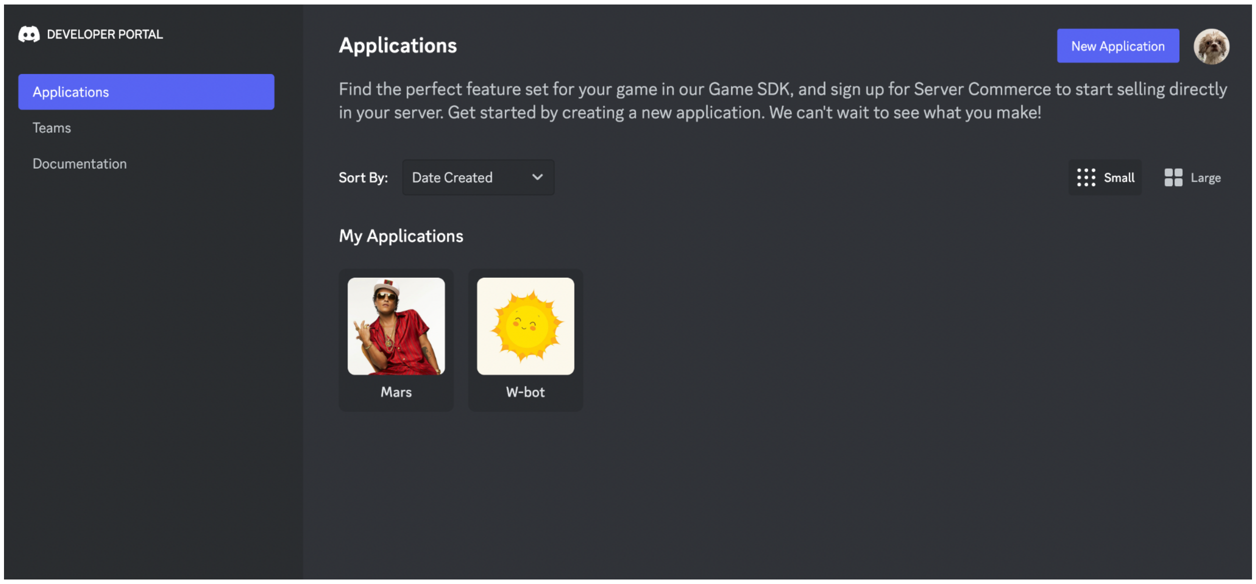
How Neshua Created W-bot (aka wot)



  - Bot Name: W-bot
  - Client ID: 1129861334993993789
  - Invite link again: https://discord.com/api/oauth2/authorize?client_id=1129861334993993789&permissions=534723951680&scope=bot%20applications.commands

Creating a Discord Application

The first step I took in creating a discord bot was to create a discord application. To do that, I simple went to the discord developer portal which looks like this:

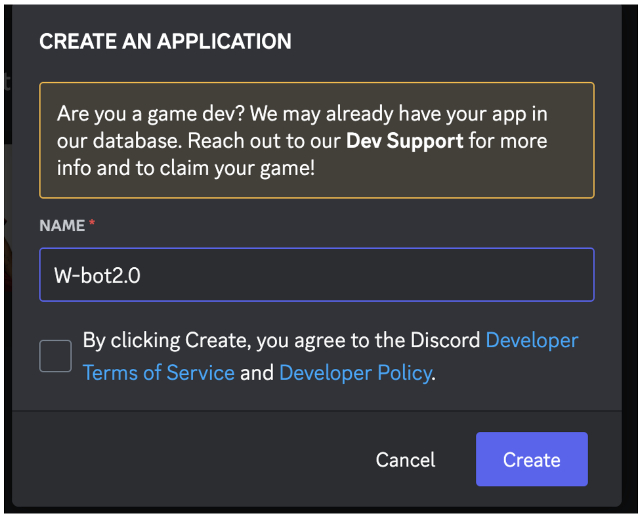


Applications

Find the perfect feature set for your game in our Game SDK, and sign up for Server Commerce to start selling directly in your server. Get started by creating a new application. We can't wait to see what you make!

My Applications

Here you can see my two and only (for now 🤞) discord bots. I clicked the new applications button on the top right which prompted this screen:



CREATE AN APPLICATION

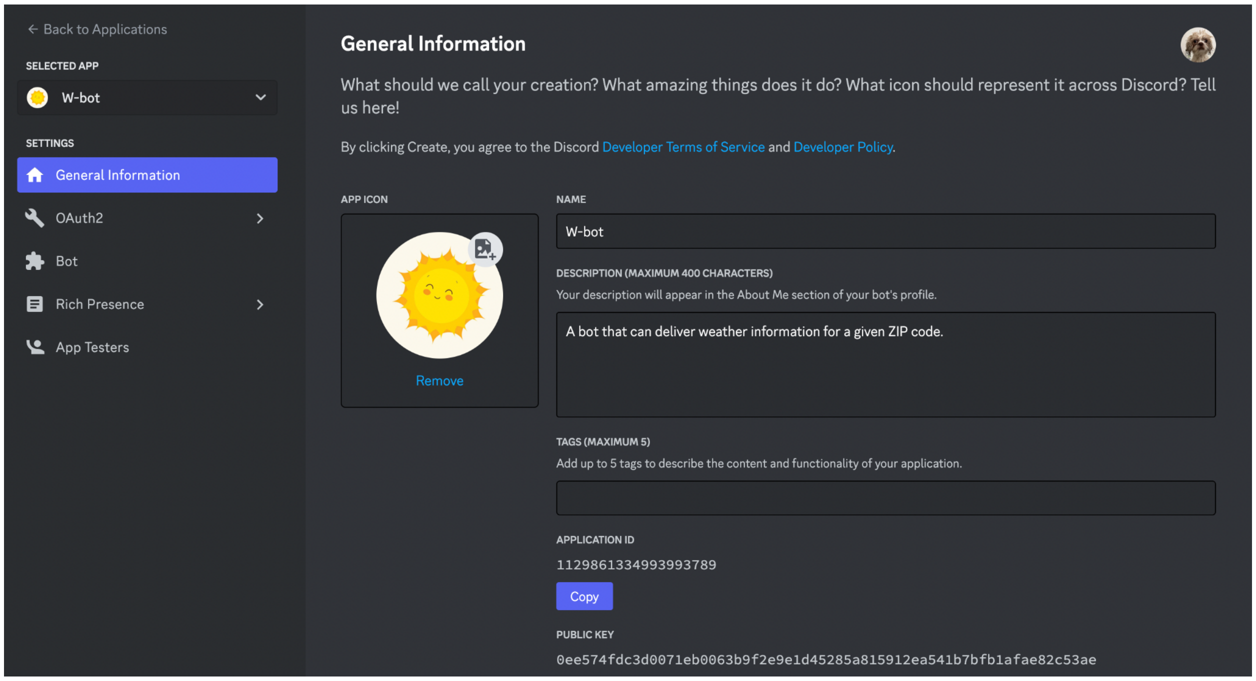
Are you a game dev? We may already have your app in our database. Reach out to our **Dev Support** for more info and to claim your game!

NAME *

W-bot2.0

By clicking Create, you agree to the Discord Developer Terms of Service and Developer Policy.

Here, I named my bot, agreed to discord's terms of service and my application was created! 🎉
This then redirected me to a General Information page for the application I just created.



General Information

What should we call your creation? What amazing things does it do? What icon should represent it across Discord? Tell us here!

By clicking Create, you agree to the Discord Developer Terms of Service and Developer Policy.

APP ICON

NAME
W-bot

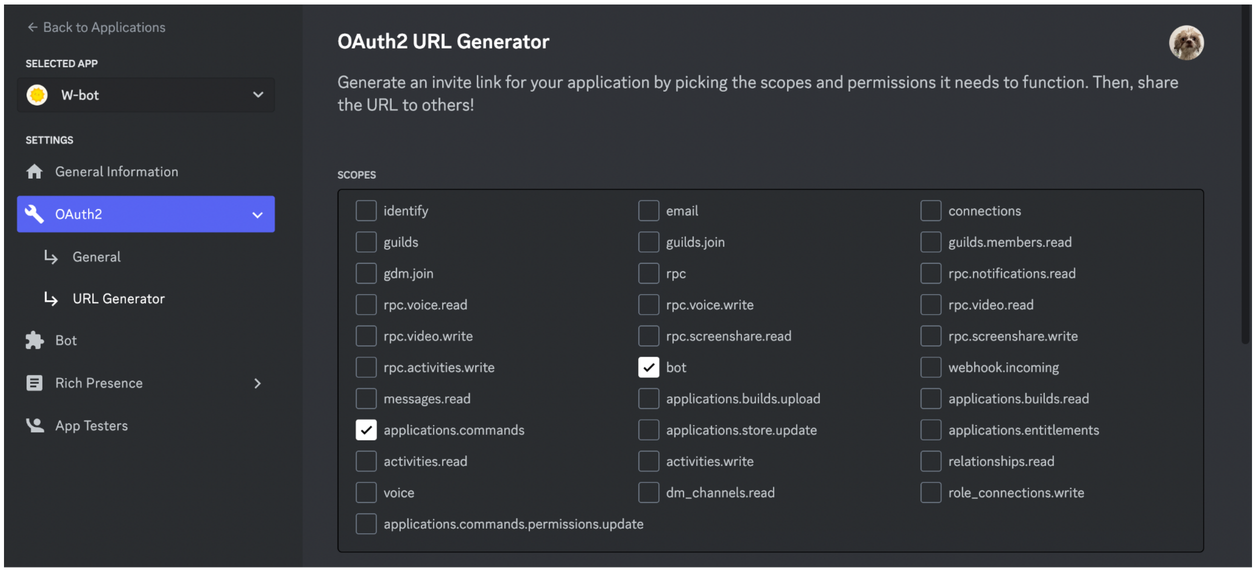
DESCRIPTION (MAXIMUM 400 CHARACTERS)
Your description will appear in the About Me section of your bot's profile.
A bot that can deliver weather information for a given ZIP code.

TAGS (MAXIMUM 5)
Add up to 5 tags to describe the content and functionality of your application.

APPLICATION ID
1129861334993993789

PUBLIC KEY
0ee574fdc3d0071eb0063b9f2e9e1d45285a815912ea541b7bfb1afae82c53ae

This is what W-bot's general information's page looks like. Here, I customized my app's icon and description to help users better understand what my bot does. In my case, w-bot delivers weather information for a given zip code. From the menu on the left hand side of the screen, I clicked on OAuth2 → URL Generator and created an invite link so that others could add W-bot into their servers.



OAuth2 URL Generator

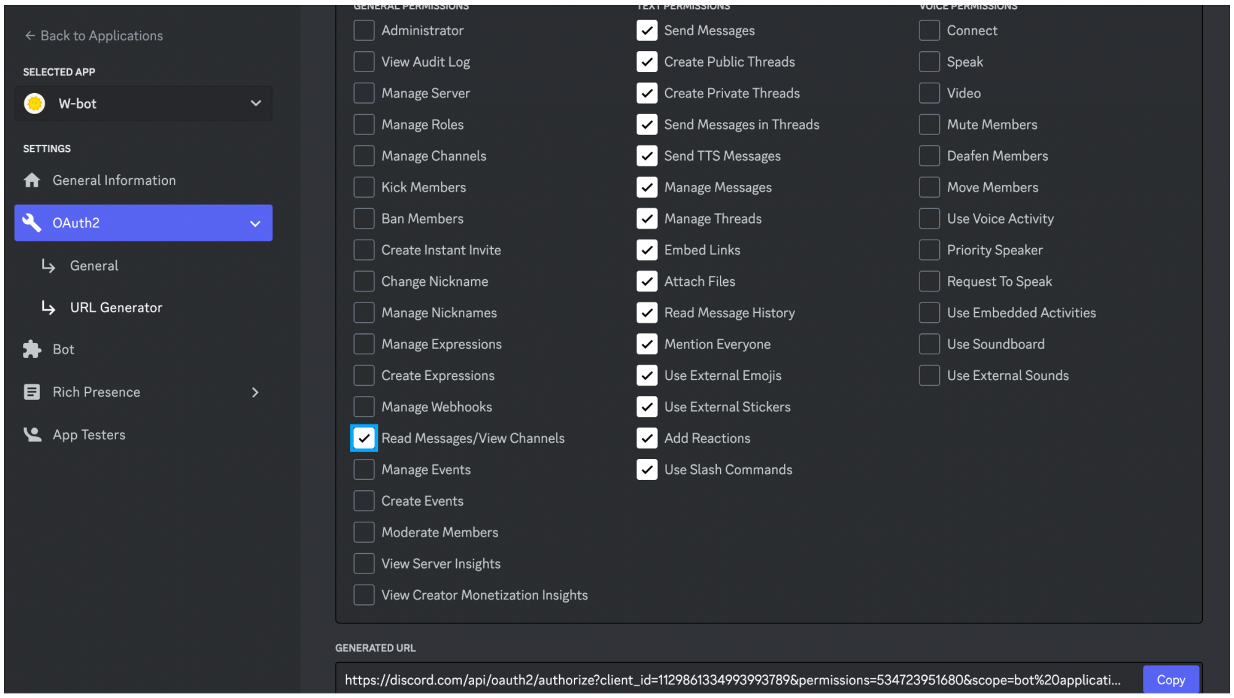
Generate an invite link for your application by picking the scopes and permissions it needs to function. Then, share the URL to others!

SCOPES

☐ identify	☐ email	☐ connections
☐ guilds	☐ guilds.join	☐ guilds.members.read
☐ gdm.join	☐ rpc	☐ rpc.notifications.read
☐ rpc.voice.read	☐ rpc.voice.write	☐ rpc.video.read
☐ rpc.video.write	☐ rpc.screenshare.read	☐ rpc.screenshare.write
☐ rpc.activities.write	✓ bot	☐ webhook.incoming
☐ messages.read	☐ applications.builds.upload	☐ applications.builds.read
✓ applications.commands	☐ applications.store.update	☐ applications.entitlements
☐ activities.read	☐ activities.write	☐ relationships.read
☐ voice	☐ dm_channels.read	☐ role_connections.write
☐ applications.commands.permissions.update		

I specified my scopes…

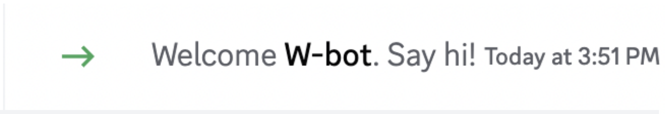
And my bot permissions:



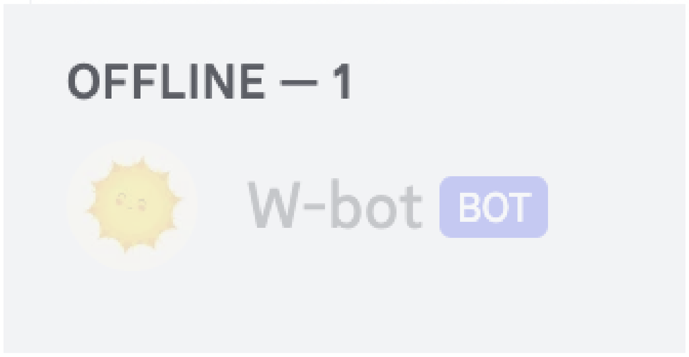
GENERAL PERMISSIONS	TEXT PERMISSIONS	VOICE PERMISSIONS
☐ Administrator	✓ Send Messages	☐ Connect
☐ View Audit Log	✓ Create Public Threads	☐ Speak
☐ Manage Server	✓ Create Private Threads	☐ Video
☐ Manage Roles	✓ Send Messages in Threads	☐ Mute Members
☐ Manage Channels	✓ Send TTS Messages	☐ Deafen Members
☐ Kick Members	✓ Manage Messages	☐ Move Members
☐ Ban Members	✓ Manage Threads	☐ Use Voice Activity
☐ Create Instant Invite	✓ Embed Links	☐ Priority Speaker
☐ Change Nickname	✓ Attach Files	☐ Request To Speak
☐ Manage Nicknames	✓ Read Message History	☐ Use Embedded Activities
☐ Manage Expressions	✓ Mention Everyone	☐ Use Soundboard
☐ Create Expressions	✓ Use External Emojis	☐ Use External Sounds
☐ Manage Webhooks	✓ Use External Stickers	
✓ Read Messages/View Channels	✓ Add Reactions	
☐ Manage Events	✓ Use Slash Commands	
☐ Create Events		
☐ Moderate Members		
☐ View Server Insights		
☐ View Creator Monetization Insights		

GENERATED URL

https://discord.com/api/oauth2/authorize?client_id=1129861334993993789&permissions=534723951680&scope=bot%20applicati...



→ Welcome **W-bot**. Say hi! Today at 3:51 PM



OFFLINE — 1

W-bot BOT

W-bot was in my server but it was offline! 😰

That's because I hadn't coded anything yet. Before starting to code, make sure to go back to your application's site and have the appropriate Privilege Gateway Intents toggled (I learned that the tough way 🥲). Now let's look at the api's I used to create this bot.

Weather API

The weather api I used for w-bot is called Open-Meteo. This api does not require an API key and has a lot of cool features. The ones I was interested in using were the current weather and forecast options.

Gif API

The gif api I used for w-bot was Giphy. This one requires an account with giphy and an api key but it is otherwise free to use.

Python

The first step in coding this bot was downloading the discord.py library to my device. I ran the following command in my console:

```
pip install discord.py
```

There are 6 python files that help create w-bot:

- Main.py ← Where the bot runs from
- Wbot.py ← This is where all the bot commands live
- Weather_gif.py ← Helper class that makes the giphy api calls to get the gif url
- weatherApi.py ← Helper class that makes open-meteo calls to get weather data
- Wmo_codes.py ← a dictionary with all World Meteorological Organization weather codes and their description

Weather API Class

This particular weather api uses longitude and latitude numbers to present weather data. We want our bot users to be presented weather data from a particular zip code which is easier to make sense of as opposed to longitude and latitude numbers (I personally don't even know how those work 😬). So my first concern was transforming a given zip code into latitude and longitude numbers. Luckily, python has this really neat library, pgeocode, that allows us to get geolocation data from a zip code. This data includes latitude and longitude! 😻

After installing and importing pgeocode to my class, I was able to get started with creating the helper functions for obtaining weather data. Here are the functions:

This `get_curr_weather(zipcode)` function takes in a zipcode and returns the current weather information from the open-meteo API

```
import pgeocode
import requests


def get_curr_weather(zipcode):
   nomi = pgeocode.Nominatim('us')
   gps_info = nomi.query_postal_code(zipcode)
   lat = gps_info['latitude']
   lon = gps_info['longitude']
   w_response =
requests.get(f'https://api.open-meteo.com/v1/forecast?latitude={lat}&
longitude={lon}&hourly'

'=temperature_2m&current_weather=true&timezone=America%2FChicago')
   weather_info = w_response.json()['current_weather']
   return weather_info
```

This function allows us to extract the following current weather information:

```
"current_weather": {
       "time": "2022-01-01T15:00"
       "temperature": 2.4, "weathercode": 3,
       "windspeed": 11.9, "winddirection": 95.0,
},
```

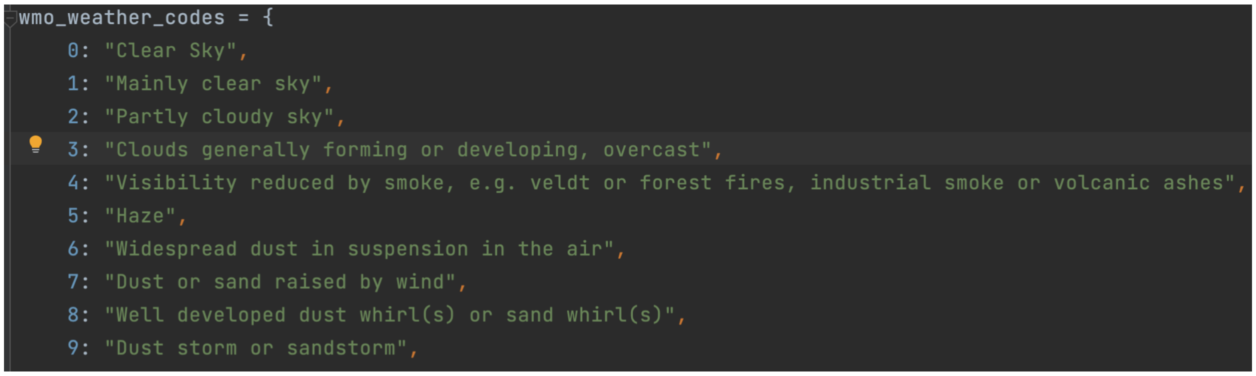
We could access specific attributes in the weather data like this:

```
get_curr_weather(zipcode)['temperature']
```

However for the sake of simplicity, and just personal preference, I created separate functions to access these specific data points.
Here's an example of a function for getting the current temperature from a zip code:

```
def get_curr_temp(zipcode):
   weather_info = get_curr_weather(zipcode)
   curr_temp = weather_info['temperature']
   return curr_temp
```

When looking at the `current_weather` data, I noticed an attribute that I couldn't quite recognize, "`weathercode`", which open-meteo's website describes as "weather condition as a numeric code." Each numeric code is associated with a weather description. I figured this would be helpful in the future when I try to look up gifs that match the reported weather so it would be wise to keep track of them. I made a dictionary for every weather code:



```
wmo_weather_codes = {
    0: "Clear Sky",
    1: "Mainly clear sky",
    2: "Partly cloudy sky",
    3: "Clouds generally forming or developing, overcast",
    4: "Visibility reduced by smoke, e.g. veldt or forest fires, industrial smoke or volcanic ashes",
    5: "Haze",
    6: "Widespread dust in suspension in the air",
    7: "Dust or sand raised by wind",
    8: "Well developed dust whirl(s) or sand whirl(s)",
    9: "Dust storm or sandstorm",
```

I also made a function that returns the current weather code for a specific zip code:

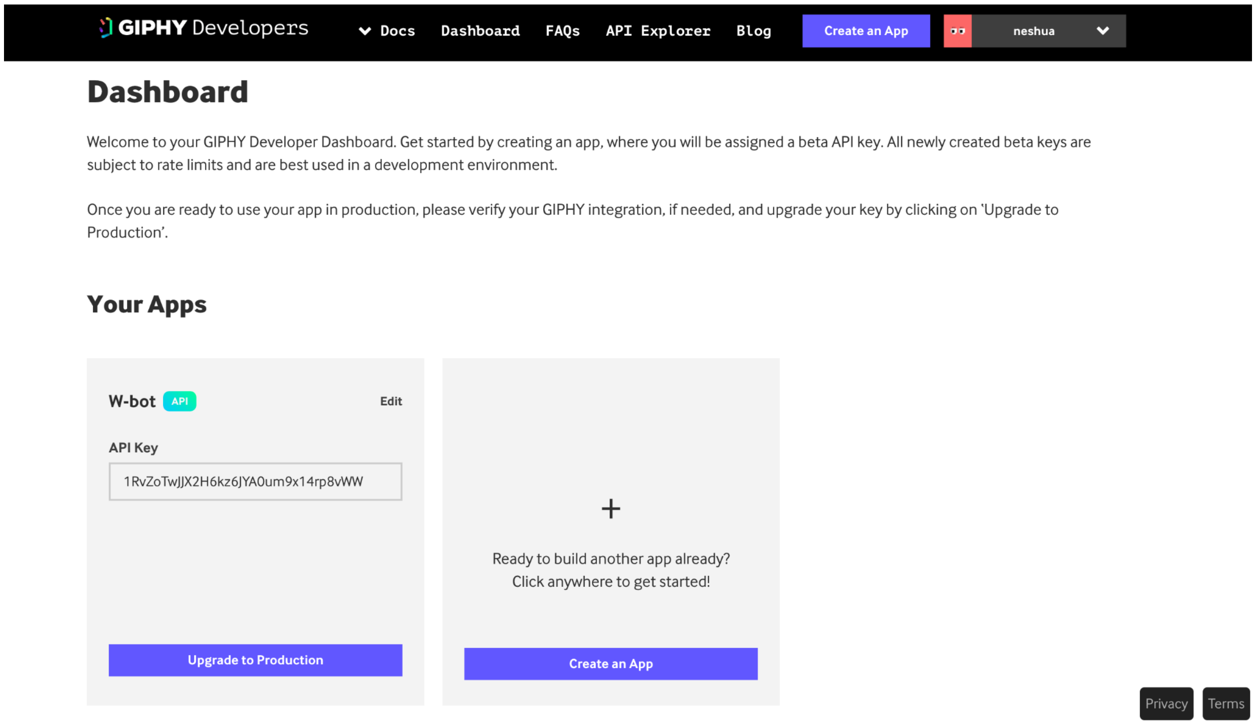
```
def get_curr_weather_code(zipcode):
    weather_info = get_curr_weather(zipcode)
    curr_weather_code = weather_info['weathercode']
    return curr_weather_code
```

In total, there are six functions that make up the `weatherApi.py` class:

- `get_curr_weather(zipcode):` extracts specific current weather information from zip code
- `get_curr_temp(zipcode):` returns the temperature in C from zip code
- `get_weather_on(zipcode, date_time):` returns weather data from zip code on specified date and time
- `get_location_name(zipcode):` returns location name from zip code
- `get_curr_weather_code(zipcode):` returns current weather code from zip code
- `get_daily_weather_code(zip,day):` returns weather code for specific date and zip code

Gif API Class

The API I used to get gifs from a given weather code was the Giphy API. This api required me to make a developer account and an application. Here you can see the W-bot app and the API key:



Dashboard

Welcome to your GIPHY Developer Dashboard. Get started by creating an app, where you will be assigned a beta API key. All newly created beta keys are subject to rate limits and are best used in a development environment.

Once you are ready to use your app in production, please verify your GIPHY integration, if needed, and upgrade your key by clicking on 'Upgrade to Production'.

Your Apps

W-bot API Edit

API Key

1RvZoTwJJX2H6kz6JYA0um9x14rp8vWW

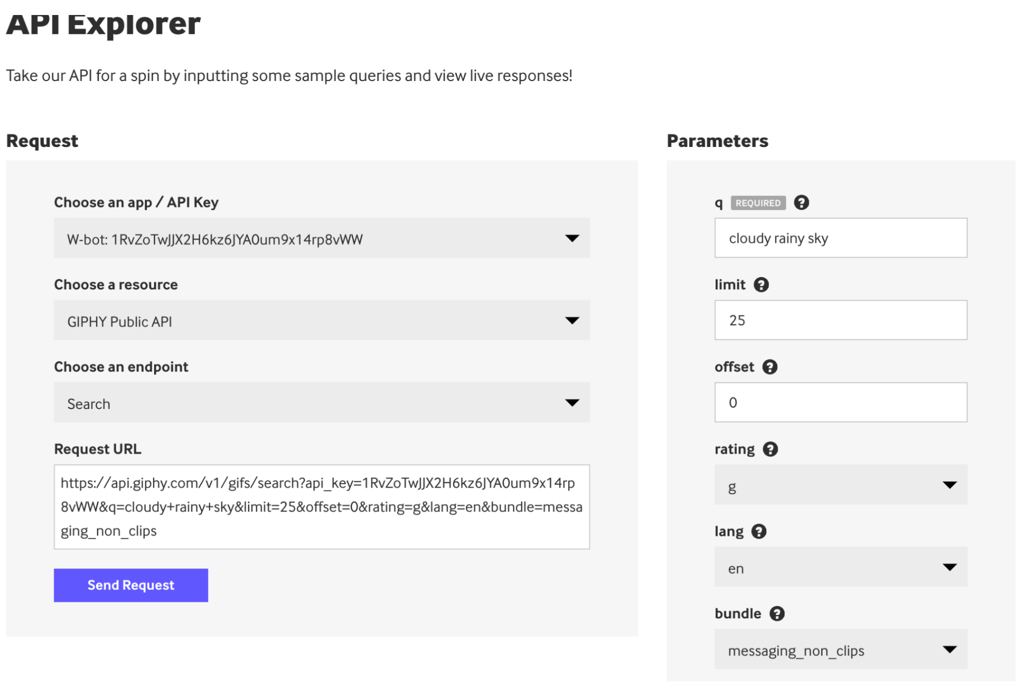
Upgrade to Production

+

Ready to build another app already?
Click anywhere to get started!

Create an App

Their API explorer page has a very easy to use GUI to create a request URL:



API Explorer

Take our API for a spin by inputting some sample queries and view live responses!

Request

Choose an app / API Key: W-bot: 1RvZoTwJJX2H6kz6JYA0um9x14rp8vWW

Choose a resource: GIPHY Public API

Choose an endpoint: Search

Request URL:

```
https://api.giphy.com/v1/gifs/search?api_key=1RvZoTwJJX2H6kz6JYA0um9x14rp8vWW&q=cloudy+rainy+sky&limit=25&offset=0&rating=g&lang=en&bundle=messaging_non_clips
```

Send Request

Parameters

q REQUIRED: cloudy rainy sky

limit: 25

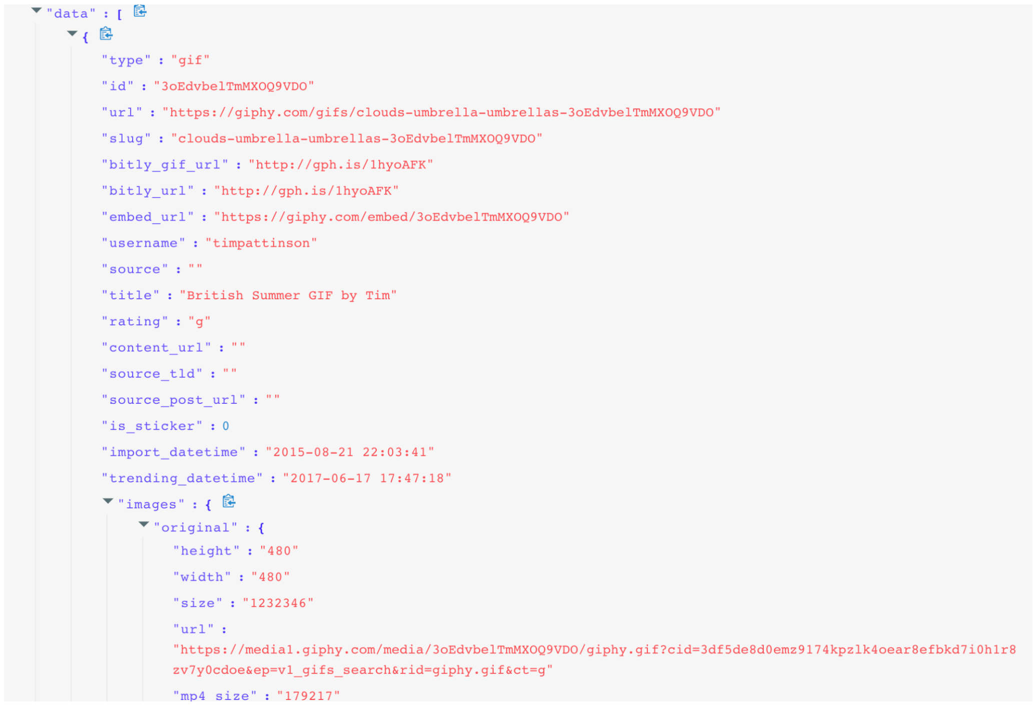
offset: 0

rating: g

lang: en

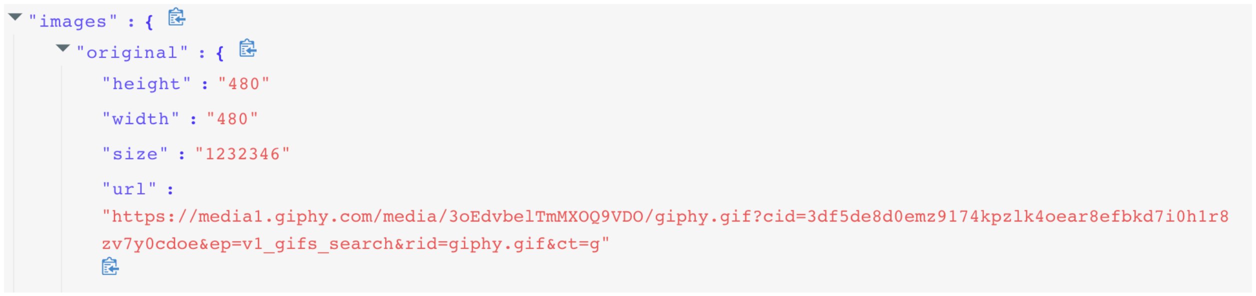
bundle: messaging_non_clips

The endpoint I am interested in is “Search”. I wrote down a simple query “cloudy rainy day” to find the gifs of a cloudy rainy day, and the default gif limit is 25. After clicking the send request button I get something that looks like this:



```
▼ "data" : [
  ▼ {
      "type" : "gif"
      "id" : "3oEdvbelTmMXOQ9VDO"
      "url" : "https://giphy.com/gifs/clouds-umbrella-umbrellas-3oEdvbelTmMXOQ9VDO"
      "slug" : "clouds-umbrella-umbrellas-3oEdvbelTmMXOQ9VDO"
      "bitly_gif_url" : "http://gph.is/1hyoAFK"
      "bitly_url" : "http://gph.is/1hyoAFK"
      "embed_url" : "https://giphy.com/embed/3oEdvbelTmMXOQ9VDO"
      "username" : "timpattinson"
      "source" : ""
      "title" : "British Summer GIF by Tim"
      "rating" : "g"
      "content_url" : ""
      "source_tld" : ""
      "source_post_url" : ""
      "is_sticker" : 0
      "import_datetime" : "2015-08-21 22:03:41"
      "trending_datetime" : "2017-06-17 17:47:18"
    ▼ "images" : {
      ▼ "original" : {
          "height" : "480"
          "width" : "480"
          "size" : "1232346"
          "url" :
          "https://media1.giphy.com/media/3oEdvbelTmMXOQ9VDO/giphy.gif?cid=3df5de8d0emz9174kpzlk4oear8efbkd7i0h1r8
          zv7y0cdoe&ep=v1_gifs_search&rid=giphy.gif&ct=g"
          "mp4_size" : "179217"
```

Out of all of this information, the image URL is what I need:



```
▼ "images" : {
  ▼ "original" : {
      "height" : "480"
      "width" : "480"
      "size" : "1232346"
      "url" :
      "https://media1.giphy.com/media/3oEdvbelTmMXOQ9VDO/giphy.gif?cid=3df5de8d0emz9174kpzlk4oear8efbkd7i0h1r8
      zv7y0cdoe&ep=v1_gifs_search&rid=giphy.gif&ct=g"
```

The API response gives us 25 gifs that match our “cloudy rainy sky” query. However, w-bot is not really interested in showing users all 25 of these gifs 😅. Because of this, when I created the helper function that returns a gif URL from the api request, I made it so it randomly picked 1 out of the 25 available gifs. This is what the `weather_gif.py` class and `get_gif()` function looks like:

```
import random
from weatherApi import *


def get_gif(query):
   gif_response = requests.get(

f"https://api.giphy.com/v1/gifs/search?api_key=1RvZoTwJJX2H6kz6JYA0um
9x14rp8vWW&q={query}&limit=25&offset=0"
       f"&rating=g&lang=en&bundle=messaging_non_clips")
   gif_info = gif_response.json()
   random_gif = random.choice(gif_info['data'])
   return random_gif['images']['original']['url']
```

W-bot Class

So far, we have a helper class to handle our weather API calls ✅and a helper class to handle our gif API call ✅, we are missing an actual bot class that will bring W-bot to life! (bit-ly speaking 🤖= 🧠🙅).

The first 10 lines of our wbot.py class look like this:

```
import discord
from wmo_codes import *
from discord import app_commands
from discord.ext import commands
from weather_gif import *

intents = discord.Intents.all()
bot = commands.Bot(command_prefix="!", intents=intents)
temp = "c"
location = 61073
```

I imported all the necessary libraries and classes I needed to create my slash commands, initiated my discord bot and set up my global default variables for temperature unit and location.

Next I set up my on_ready() bot event:

```
@bot.event
async def on_ready():
   print("W-bot is ready!")
   try:
       synced = await bot.tree.sync()
       print(f"Synced {len(synced)} command(s)")
   except Exception as e:
       print(e)
```

After this bot event, I created the `/set_location` and `/set_temp_unit` commands which update the global location and global temp variables respectively. Now, when the user calls these commands they will be able to set the temperature unit to either celsius or fahrenheit and specify their preferred location's zip code.

This is the code for these commands:

```
@bot.tree.command(name="set_temp_unit", description="Set your
preferred temperature unit")
@app_commands.describe(temp_unit="Type: 'c' or 'f'")
async def set_temp_unit(interaction: discord.Interaction, temp_unit:
str):
   global temp
   temp = temp_unit.lower()
   await interaction.response.send_message(f"Temperature unit set to
{temp}")


@bot.tree.command(name="set_location", description="Set where you'd
like to get weather information from")
@app_commands.describe(zipcode="Enter the zipcode of your desired
location")
async def set_location(interaction: discord.Interaction, zipcode:
int):
   global location
   location = zipcode
   await interaction.response.send_message(f"Location set to " +
get_location_name(zipcode))
```

This bot is meant to help friends who'd like to pick the perfect weather day to meet up, hence, I created the command /`get_temp_on` that allows users to look up the temperature for a specific date and time. This command takes in a date and time in the YYYY-MM-DDT00:00 format. It also embeds a gif for the appropriate weather code.

```
@bot.tree.command(name="get_temp_on", description="Get the
temperature on a specific day and time(7 days from today)")
@app_commands.describe(date_time="Time and Date YYYY-MM-DDT00:00
format ex.2023-07-18T15:00")
async def get_temp_on(interaction: discord.Interaction, date_time:
str):
   global temp
   global location
   date = date_time[:10]
   time = date_time[11:]
   embed = discord.Embed()

embed.set_image(url=get_gif(wmo_weather_codes[get_daily_weather_code(
location, date)]))
   if temp == "f":
       await interaction.response.send_message(f"The temperature in "
+ get_location_name(location) + " will be: \n" +


str((get_weather_on(location, date_time)*9/5)+32) + f" F on {date} at
{time} "
                                               , embed=embed)
   else:
       await interaction.response.send_message(f"The temperature in "
+ get_location_name(location) + "will be: \n" +

str(get_weather_on(location, date_time)) + f" F on {date} at {time}
",
                                               embed=embed)
```

This part of the code:

```
embed.set_image(url=get_gif(wmo_weather_codes[get_daily_weather_code(l
ocation, date)]))
```

Uses the `get_gif()` function I mentioned in the Gif API Class section of this doc. The query input is the weather code specific to that location and date. Again, this code translates to a specific weather description which we can access through the `wmo_weather_codes` dictionary mentioned in the [Weather API Class section of this doc.](#)

Next, is the `/weather_info` command. This command gives the user the current temperature, wind speed and wind direction. Here is the code for that command:

```
@bot.tree.command(name="weather_info", description="Get an overview of your
location's current weather")
async def weather_info(interaction: discord.Interaction):
   global location
   global temp
   if temp == "f":
       await interaction.response.send_message(f"The current weather report
for " + get_location_name(location) +
                                               " is: \n
----------------------- \n Temperature = "
                                               + str(
           (get_curr_weather(location)['temperature'] * 9 / 5) + 32) + "
F\n -----------------------\n Wind Speed = " +

str(get_curr_weather(location)['windspeed']) +
                                               " \n -----------------------
\n Wind Direction = " + str(
           get_curr_weather(location)['winddirection']) + "\n
-----------------------")

   else:
       await interaction.response.send_message(f"The current weather report
for " + get_location_name(location) +
                                               " is: \n
----------------------- \n Temperature = "
                                               +
str(get_curr_weather(location)['temperature']) +
                                               " C\n
----------------------- \n Wind Speed = " +

str(get_curr_weather(location)['windspeed']) +
                                               " \n -----------------------
\n Wind Direction = " + str(
           get_curr_weather(location)['winddirection']) + "\n
----------------------- ")
```

Finally, we have the `/get_temp_now` command which gives users the current temperature in their preferred location and embeds a gif for the appropriate weather code:

```
@bot.tree.command(name="get_temp_now", description="Get your
location's current temperature")
async def get_temp_now(interaction: discord.Interaction):
   global location
   global temp
   embed = discord.Embed()

embed.set_image(url=get_gif(wmo_weather_codes[get_curr_weather_code(l
ocation)]))
   if temp == "f":
       await interaction.response.send_message(
           f"The current temperature in " +
get_location_name(location) + " is: \n" +
           str((get_curr_temp(location) * 9 / 5) + 32) + f" F \n "
           + str(wmo_weather_codes[get_curr_weather_code(location)]),
           embed=embed)

   else:
       await interaction.response.send_message(
           f"The current temperature in " +
get_location_name(location) + " is: \n" +
           str(get_curr_temp(location)) + f" C \n" +
str(wmo_weather_codes[get_curr_weather_code(location)]),
           embed=embed)
```

Future improvements

Overall, I'd say W-bot is pretty cool 😎(😝). It allows users to:

- Swap between celsius and fahrenheit with a quick command
- Get current weather information of desired zip code(windspeed, wind direction, temperature)
- Set the preferred location through a zip code with a quick command
- Get weather information of desired date and time 7 days ahead
- Get a relevant GIF that matches the reported weather

If I were to make w-bot even cooler, I'd do the following:

- Zip codes: Currently, w-bot does not check for invalid zip codes. In the future, I'd like to implement a check that notifies users if their zip code was not valid and prevents the bot from crashing from non-existent zip codes.

- Gifs: Describing the current weather through weather codes may not be the most efficient way to look up gifs. Sometimes the recommended gifs do not match the weather appropriately. Using a weather api with more detailed weather descriptions could improve gif search accuracy.
- Date and Time: Currently, w-bot allows users to look up the weather 7 days in advance which is great for making plans! However, the date format for looking up the weather can be confusing and not intuitive. Introducing date conversion python libraries to my code could make the date inputs easier for users.
- Personalization: I'd love to be able to customize the font sizes and colors of w-bot in the future. As of now, I don't know how to "beautify" the bot command responses. I may need to do some more research on discord's developer page!
- Other cool things: It would be nice to show users a table of the weather for the next seven days instead of one specific day at a time. With this specific API, I believe this functionality would require a lot of string manipulation code which is time consuming and outside the 4hr scope for this challenge.