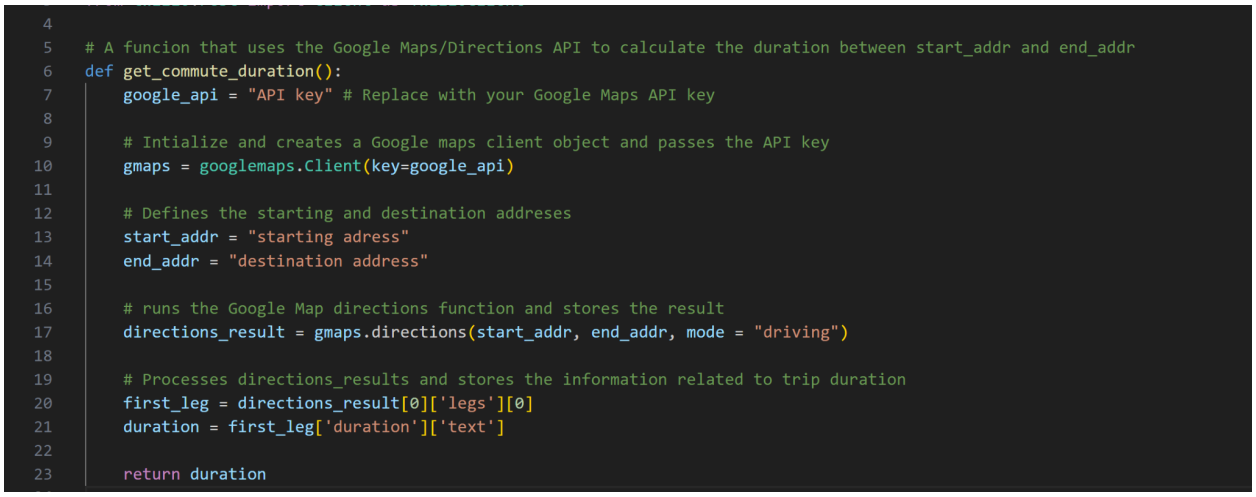
Code Overview

```
Projects > daily_commute_text.py > ...
1  import datetime
2  import googlemaps
3  from twilio.rest import Client as TwilioClient
```

Lines 1-3 import the modules datetime, googlemaps, and Client (as TwilioClient) from twilio.rest

If you need to install googlemaps and twilio.rest modules, run the following command in your terminal:

`pip install googlemaps
`pip install twilio



```
4
5  # A funcion that uses the Google Maps/Directions API to calculate the duration between start_addr and end_addr
6  def get_commute_duration():
7      google_api = "API key" # Replace with your Google Maps API key
8
9      # Intialize and creates a Google maps client object and passes the API key
10     gmaps = googlemaps.Client(key=google_api)
11
12     # Defines the starting and destination addreses
13     start_addr = "starting adress"
14     end_addr = "destination address"
15
16     # runs the Google Map directions function and stores the result
17     directions_result = gmaps.directions(start_addr, end_addr, mode = "driving")
18
19     # Processes directions_results and stores the information related to trip duration
20     first_leg = directions_result[0]['legs'][0]
21     duration = first_leg['duration']['text']
22
23     return duration
```

Lines 5 - 23 defines the get_commute_duration() function.

- Line 7 assigns our required Google Maps API key to a variable named google_api
- Line 10 initializes and creates a Google Maps client object named gmaps and passes our API key
- Line 13 - 14 defines our starting/destination addresses and stores them into variables start_addr and end_addr
- Line 17 runs the directions function and passes in start_addr, end_addr
    - The third argument is mode = “driving” to specify that the trip details should be calculated from a driving context
    - The information result is stored into a variable called directions_results
- Line 20-21 processes the resulting data and identifies/stores information related to the trip duration into a variable called duration
- Line 23 makes the function return duration

```
26
27  # A function that uses the Twilio API to create a message and send to a recipient's phone number
28  def send_text_message(message):
29      # Replace with your Twilio account SID, token, and phone number (format: "1##########"")
30      twilio_account_sid = "account sid"
31      twilio_account_token = "account token"
32      twilio_phone_num = "twilio phone number"
33
34      # Defines the recipient's phone number (format: "1##########"")
35      recipient_phone_num = "recipient phone number"
36
37      # Intializes and creates a Twilio client object and passes Twilio credentials
38      twilio_client = TwilioClient(twilio_account_sid, twilio_account_token)
39
40      # creates a Twilio message and defines it's to, from, and body data
41      twilio_client.messages.create(
42          to = recipient_phone_num,
43          from_ = twilio_phone_num,
44          body = message
45      )
46
```

Line 28 - 45 defines the send_text_message(message) function

- Line 30 - 32 creates variables called twilio_account_sid, twilio_account_token, and twilio_phone_num to store the required Twilio API information
- Line 35 defines our recipient's phone number and stores it in a variable called reciient_phone_num
- Line 38 initializes and creatives a Twilio client (called twilio_client), and passes in twilio_account_sid and twilio_account_token
- Line 41 - 45 runs the client object's messages.create function and passes in recipient_phone_num, twilio_phone_num, and message, which correspond to the to, from, and body arguments

```
49
50  duration = get_commute_duration()                    # Assigns the commute duration time between start/end addresses defined in the function
51  current_time = datetime.datetime.now()               # Assigns the current time
52
```

Line 50 calls the get_commute_duration() function and stores the returned data into a variable called duration

- The returned duration format when calling get_commute duration is: "# hours # mins"

Line 51 calls the datetime.datetime.now() function, which generates the current date and time (based on your local OS) and stores into a variable called current_time

```
52
53   # Splits up duration into individual indexes so that it can be processed
54   duration_parts = duration.split()
55
```

Line 54 runs the split() function on duration, which splits the string data into separate string chunks, which we can use to process and identify the literal hours/minutes information (trip duration)

```
55
56   #intialize values for hours, minutes to 0
57   hours, minutes = 0, 0
58
```

Line 57 initializes the values for variables hours, minutes to 0. These variables will eventually store the int value of the literal string data pertaining to trip duration information

```
59  # Processes duration_parts by identifying the string value of hour and min, and converting it to a int value to be stored in hours, minutes
60  for i in range(len(duration_parts)):
61      if duration_parts[i] == "hour":
62          hours = int(duration_parts[i - 1])
63      elif duration_parts[i] == "mins":
64          minutes = int(duration_parts[i - 1])
65
```

Line 60 - 64 process duration_parts

- The for if/else statement checks each index to see if matches “hours” or “mins”. If a match is found, the prior index should contain a number string representing hours or minutes of the trip duration
- A int function is then called at the matching index position - 1, which turns the number literal string into an int

```
65
66  # Creates a timedelta object representing the duration of commute time.
67  # commute_duration differs from duration because timedelta objects can be used to perform math operations with datetime objects
68  # whereas duration is the duration represented as a string
69  commute_duration = datetime.timedelta(hours=hours, minutes=minutes)
70
```

Line 69 uses the datetime.timedelta method and passes in the int values of hours and minutes to create a timedelta object. This timedelta object is stored into variable called commute_duration

- Timedelta objects contain time information (hours, minutes, etc)
- Mathematical operations can be performed between timedelta objects

```
70
71  # Calculates the arrival time by adding current_time (datetime object) with commute_duration (timedelta object)
72  # Formats in HH:MM AM/PM format
73  arrival_time = (current_time + commute_duration).strftime('%I:%M %p')
74
```

Line 73 adds the time information from current_time and commute_duration together, which effectively is the arrival time.

- The resulting information is formatted to HH:MM AM/PM format by calling the .strftime() method and passing in '%I:%M %p'

```
74
75  # Creates the text body message to be stored in variable message, which will be used as an argument in send_text_message function
76  message = (
77      f"\n\nGood morning!\n\n"
78      f"This is your daily morning commute forecast. \n\n"
79      f"The estimated commute time is: {duration}.\n"
80      f"If you leave now, your estimated arrival time is: {arrival_time}."
81  )
```

Line 76 creates a string message and stores it in variable named message

- The string contains data from variables duration and arrival_time

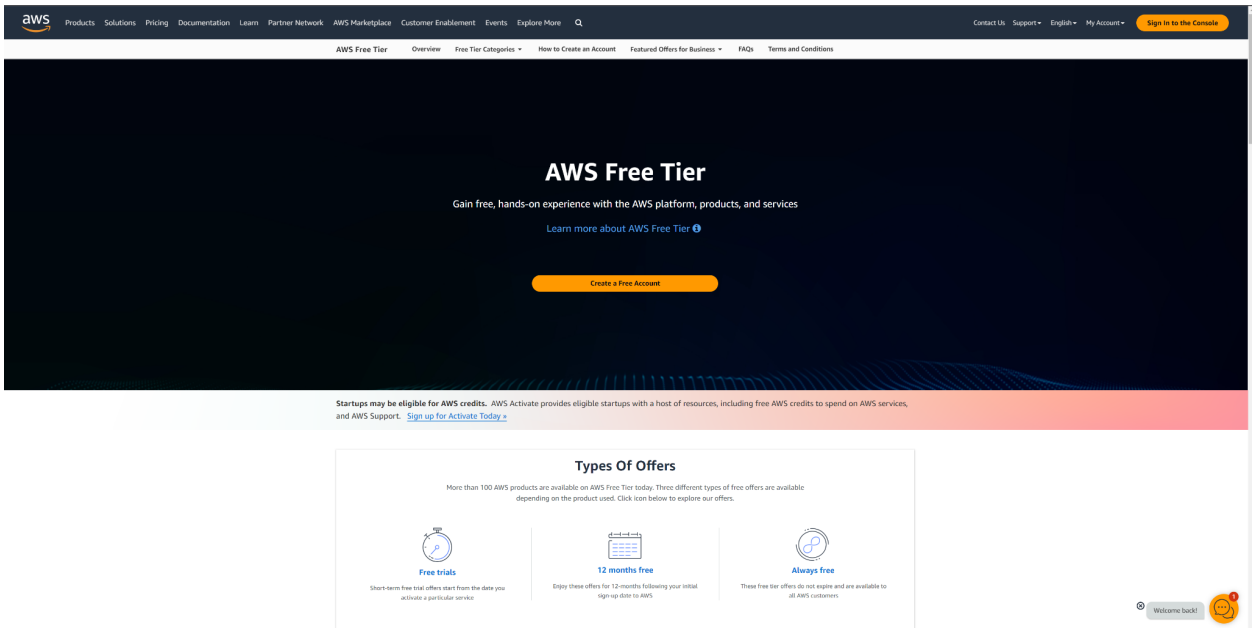
```
83
84  # Calls the send_text_message function
85  send_text_message(message)
86
```

Line 85 calls the send_text_message and passes in the variable message as an argument

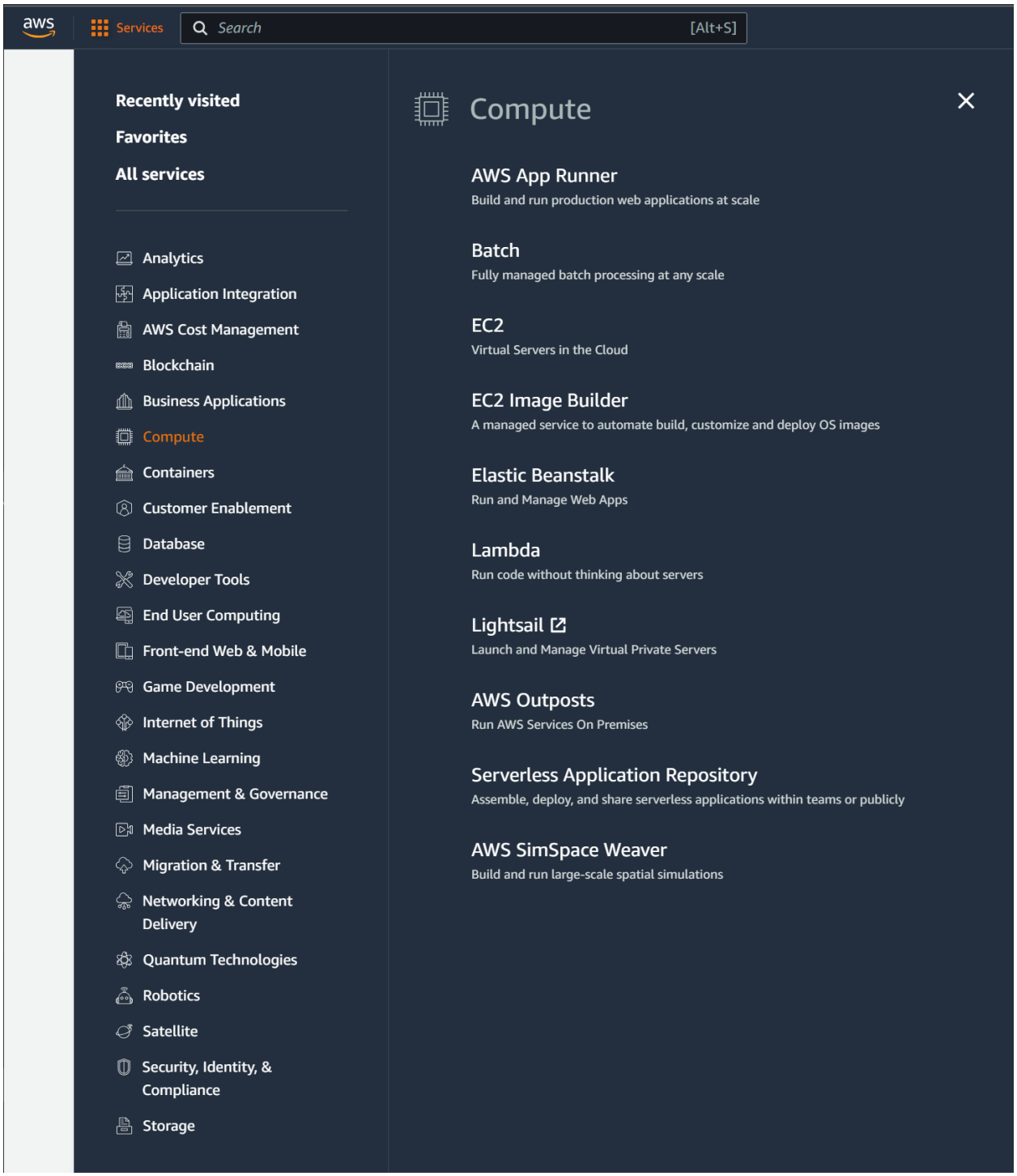
AWS Deployment/Cron Job Automation

Setting Up and Launching EC2 Instance

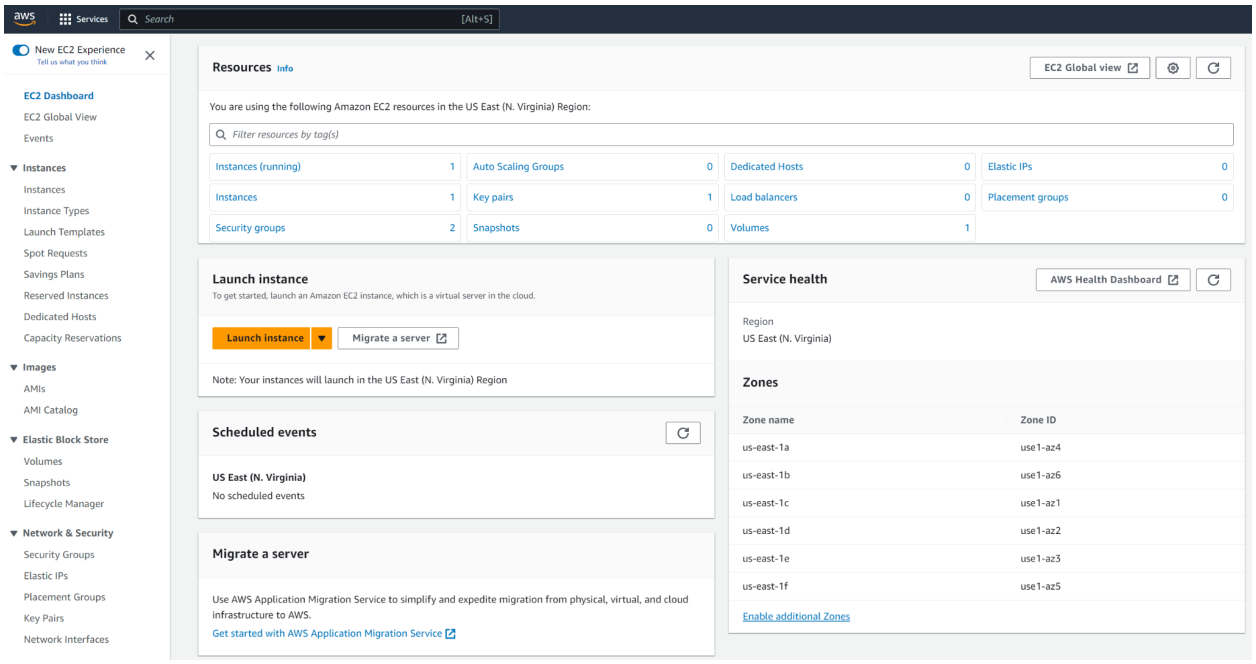
Sign into AWS. If needed, register for a free tier account.



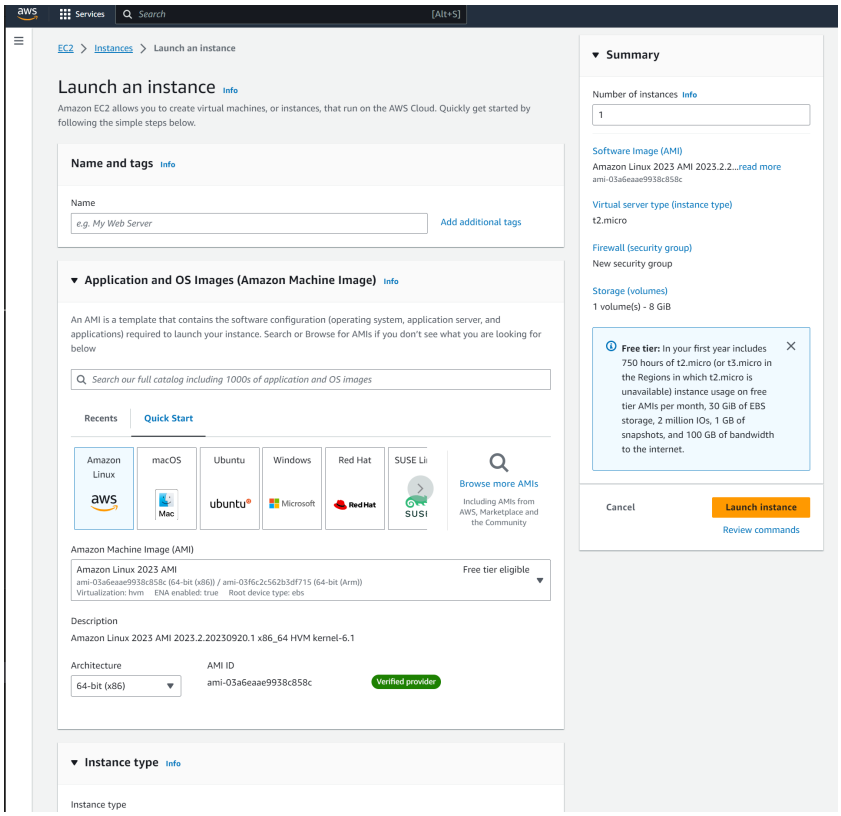
From your main dashboard, go to Services > Compute > EC2



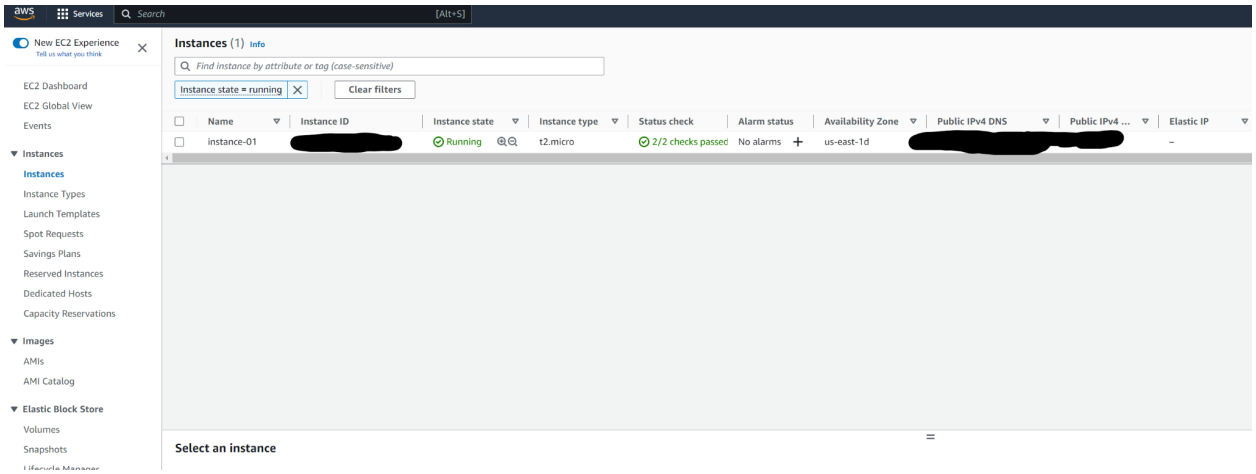
Click on Launch Instance



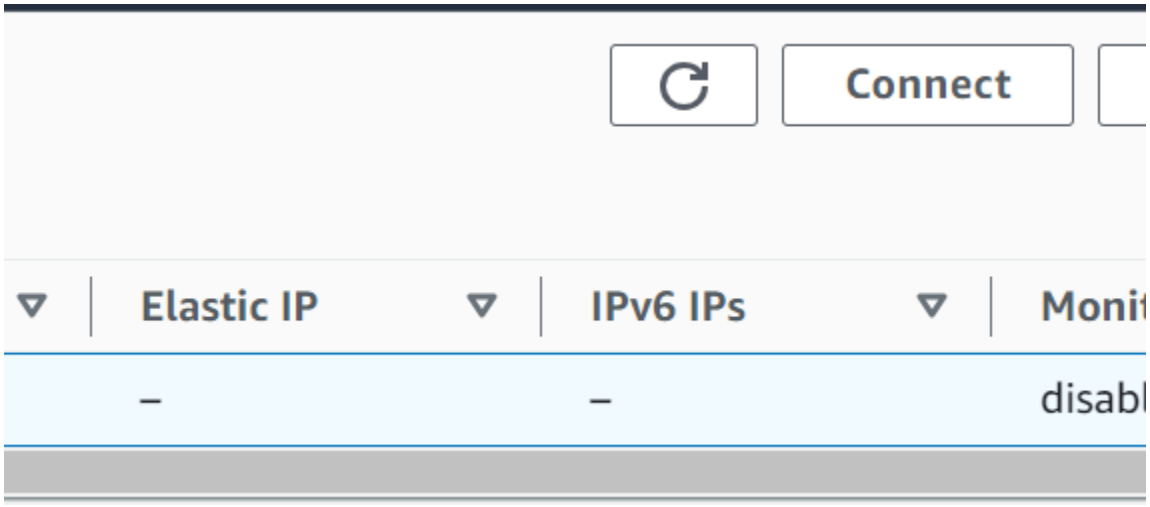
Fill out the instance information with the image being Ubuntu. Be sure to create an SSH pair if you want to connect with a client (i.e. Putty, etc)



Once you create the instance, you will see the created instance on the Instances page. It may take a few minutes for the Instance State to change to Running



Check your instance, and click on Connect. If you wish to connect via your own SSH client instead, copy and paste the instance's IP address and upload the SSH Key file accordingly



Ubuntu Terminal

Once you are logged into your instance's terminal, you should complete any Ubuntu updates and install python3. Run the following commands in your terminal:

Update your package list:
`sudo apt update

Upgrade packages
`sudo apt upgrade

If not already installed, install Python3
`sudo apt install python3

If needed, reboot your instance manually or with command:
`sudo reboot

You will need to get a copy of your script file onto your local instance. You can achieve this by uploading your script file to your github repo and then running the following terminal command:

`wget https://github.com/<username>/<reponame>/<scriptfilename>.py

Ensure that you do not upload a script file with your API key information in it. You will get your APi key deactivated and will need to rotate keys if you do so

Ensure that script file is executable by running this command:
`sudo chmod +x <scriptfilename>.py

You will need to add your API credentials. You can do this by opening your script file using the nano command and writing it in:
`nano <scriptfilename>.py

Additionally while in nano, add the shebang/hashbang as the first line of the script file. A shebang/hashbang is needed to specify the interpreter needed to run the script, This allows the script file to be ran without explicitly calling python3 in terminal commands
`#!/usr/bin/python3

Your script file may have Windows-style line endings (CRLF), which will not allow your script to run through the shebang/hashbang. To remediate this, convert the script file to Unix-style line endings (LF):
Install dos2unix by running terminal command:
`sudo apt-get install dos2unix

Convert the script with terminal command:
`dos2unix <scriptfilename>.py

Double-check your local instance timezone by running the terminal command:
`date

If the timezone is not correct, run this terminal command to print a list of timezones:
`timedatectl list-timezones

Then run this terminal command to make the change to the correct timezone based on prior command:
`sudo timedatectl set-timezone <timezone>

Scheduling Cron Job

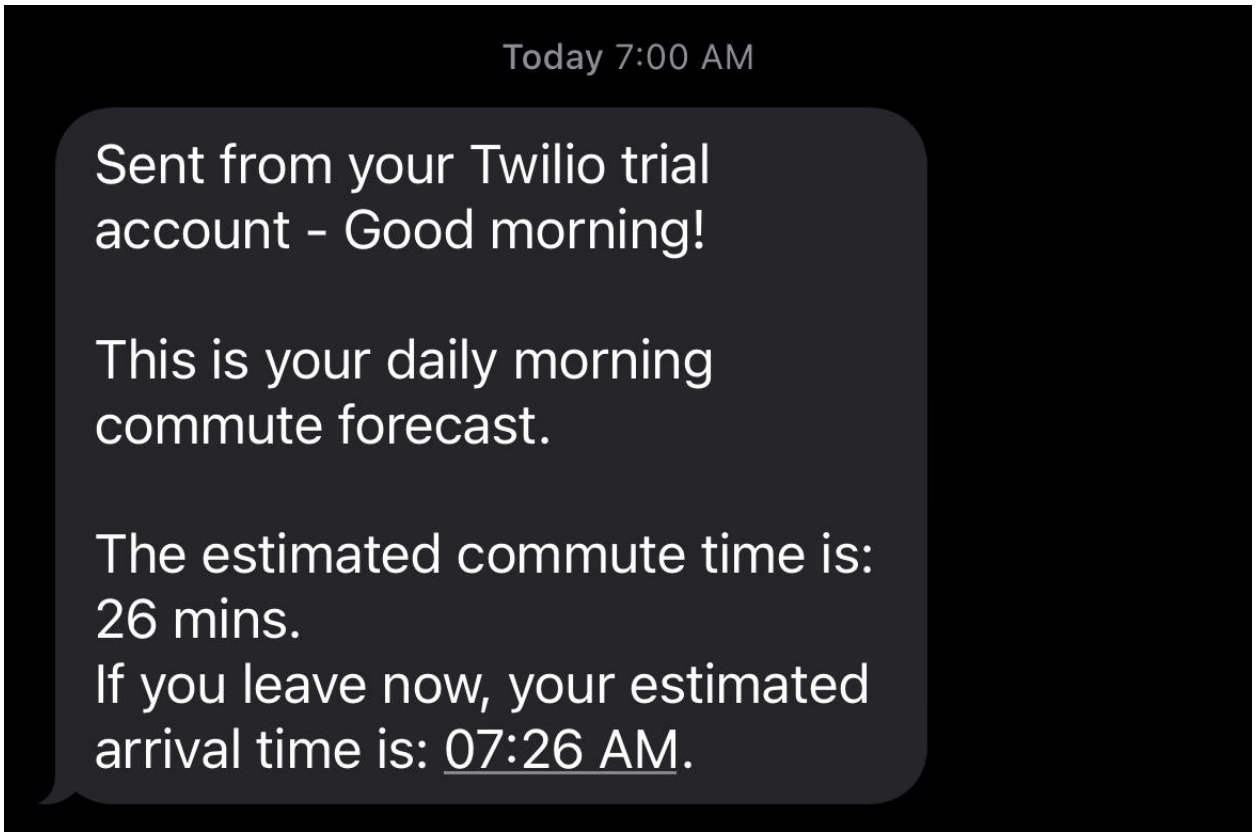
To set up a cron job, run terminal command:
`crontab -e

Please see reference material at https://www.hostinger.com/tutorials/cron-job regarding syntax and guidance

Once the cron job is scheduled, the script file will be executed at the specified schedule

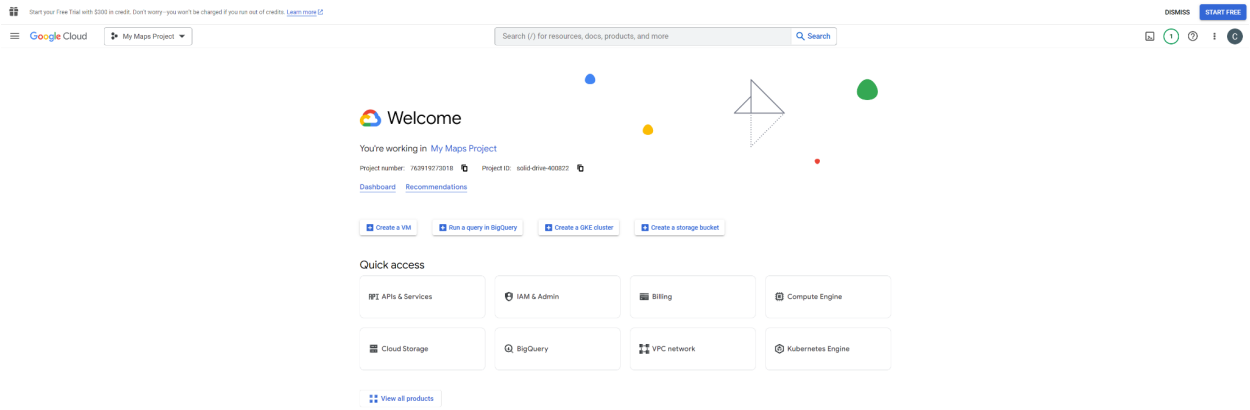
Result

Scheduled text message:

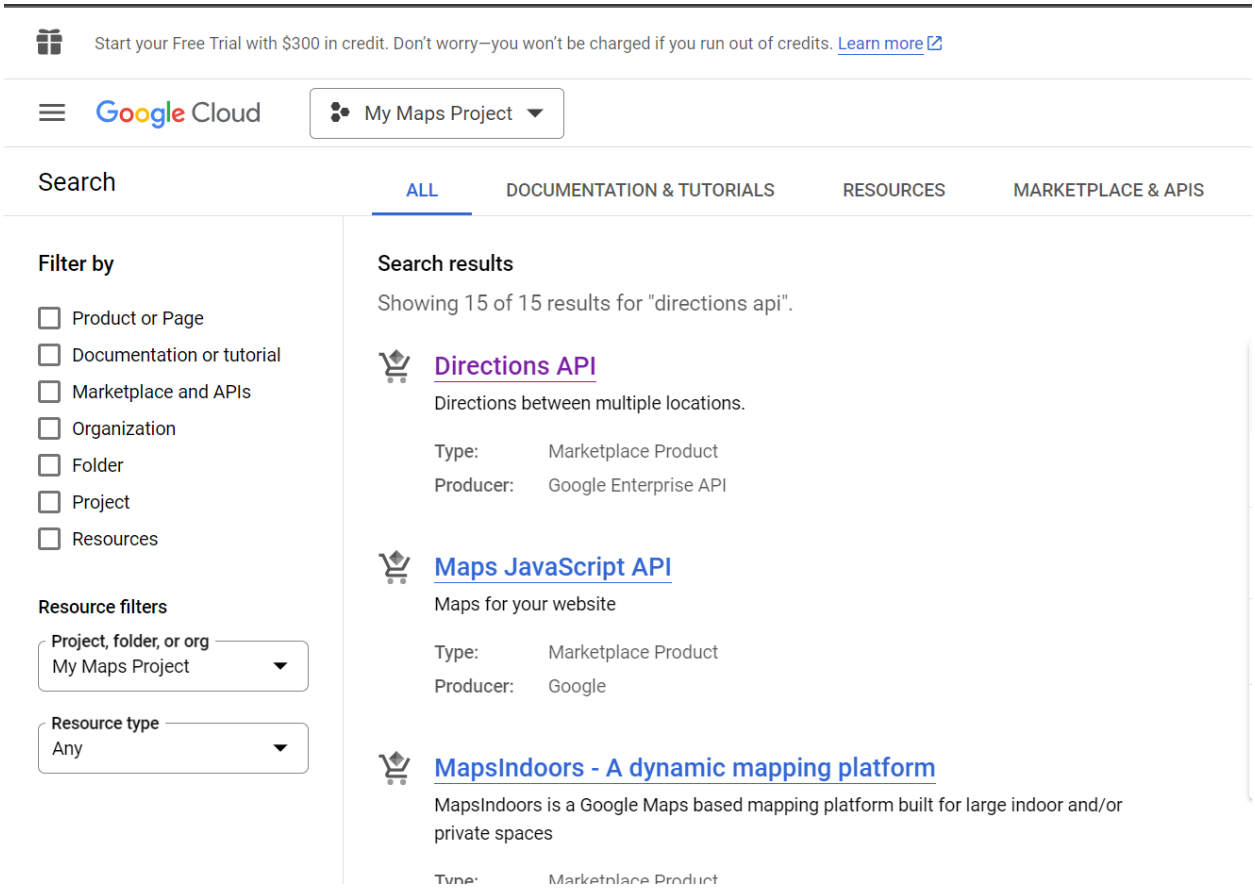


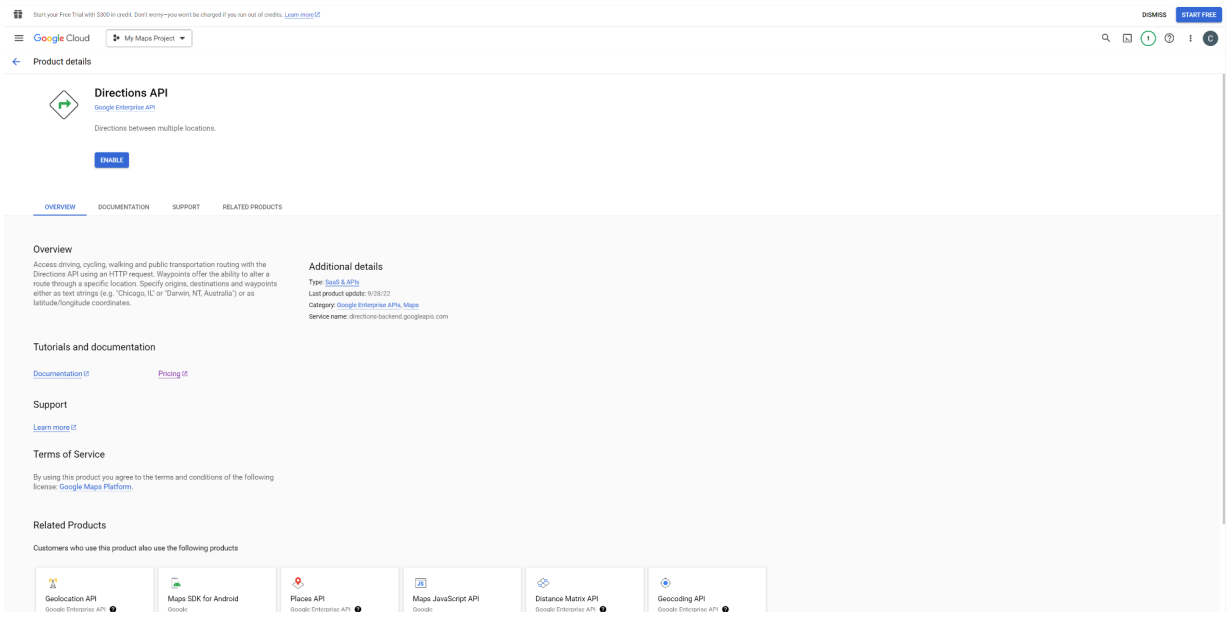
How to set up Google Maps API key

Go to https://console.google.com. You can use your gmail account or create a new one.



Search for ‘Directions API’

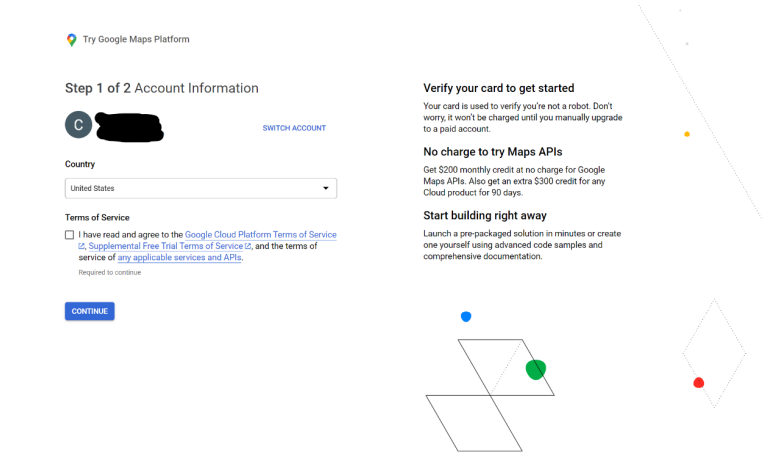

Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more
Google Cloud
My Maps Project
Search
ALL
DOCUMENTATION & TUTORIALS
RESOURCES
MARKETPLACE & APIS
Filter by
Product or Page
Documentation or tutorial
Marketplace and APIs
Organization
Folder
Project
Resources
Resource filters
Project, folder, or org
My Maps Project
Resource type
Any
Search results
Showing 15 of 15 results for "directions api".
Directions API
Directions between multiple locations.
Type: Marketplace Product
Producer: Google Enterprise API
Maps JavaScript API
Maps for your website
Type: Marketplace Product
Producer: Google
MapsIndoors - A dynamic mapping platform
MapsIndoors is a Google Maps based mapping platform built for large indoor and/or private spaces
Type: Marketplace Product

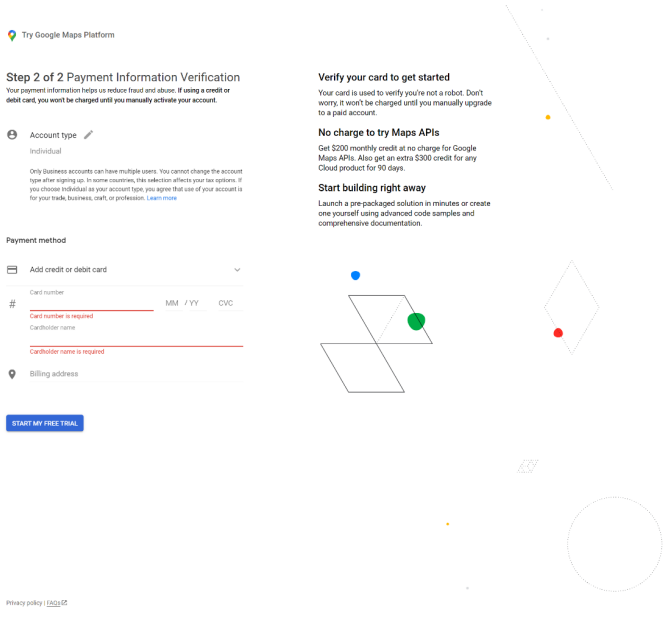

DISMISS
START FREE
Google Cloud
My Maps Project
Product details
Directions API
Google Enterprise API
Directions between multiple locations.
ENABLE
OVERVIEW
DOCUMENTATION
SUPPORT
RELATED PRODUCTS
Overview
Additional details
Tutorials and documentation
Documentation
Pricing
Support
Learn more
Terms of Service
Related Products
Customers who use this product also use the following products
Geolocation API
Maps SDK for Android
Places API
Maps JavaScript API
Distance Matrix API
Geocoding API

If you've never utilized Google Developer tools, you'll get taken to a page to confirm terms and conditions and verify billing information to activate your free trial.

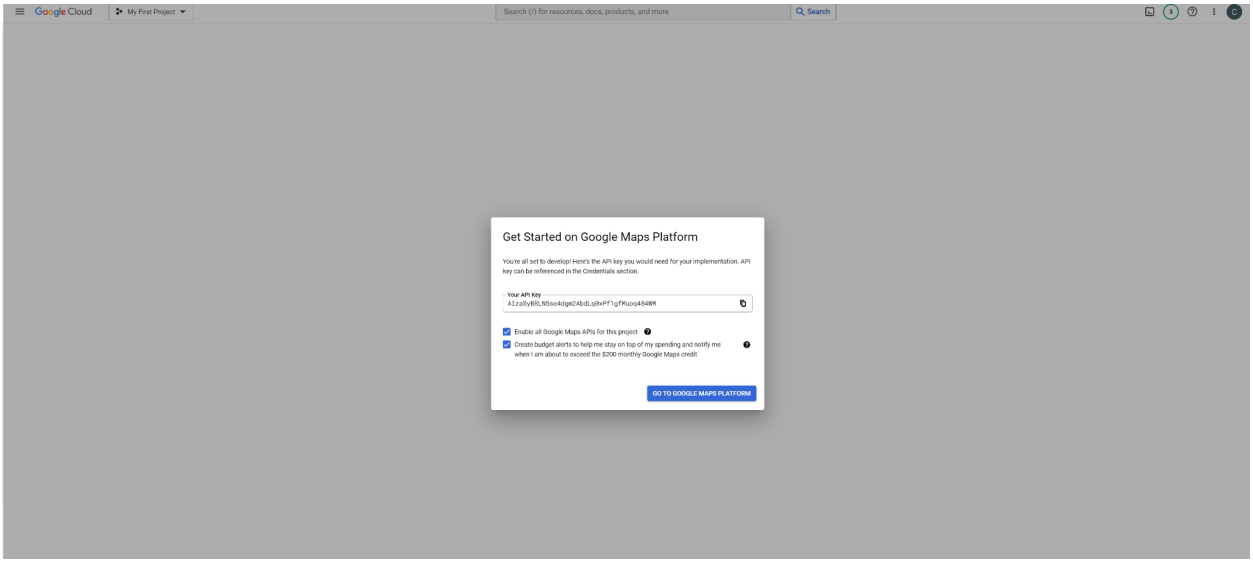
The Google Maps Platform offers $200 monthly credit at no charge for Google Maps APIs, but you should review and familiarize yourself with Google's billings and costs to avoid any unexpected charges.

You may see a charge to your credit card (for billing verification) and it should fall off your statement shortly afterwards.

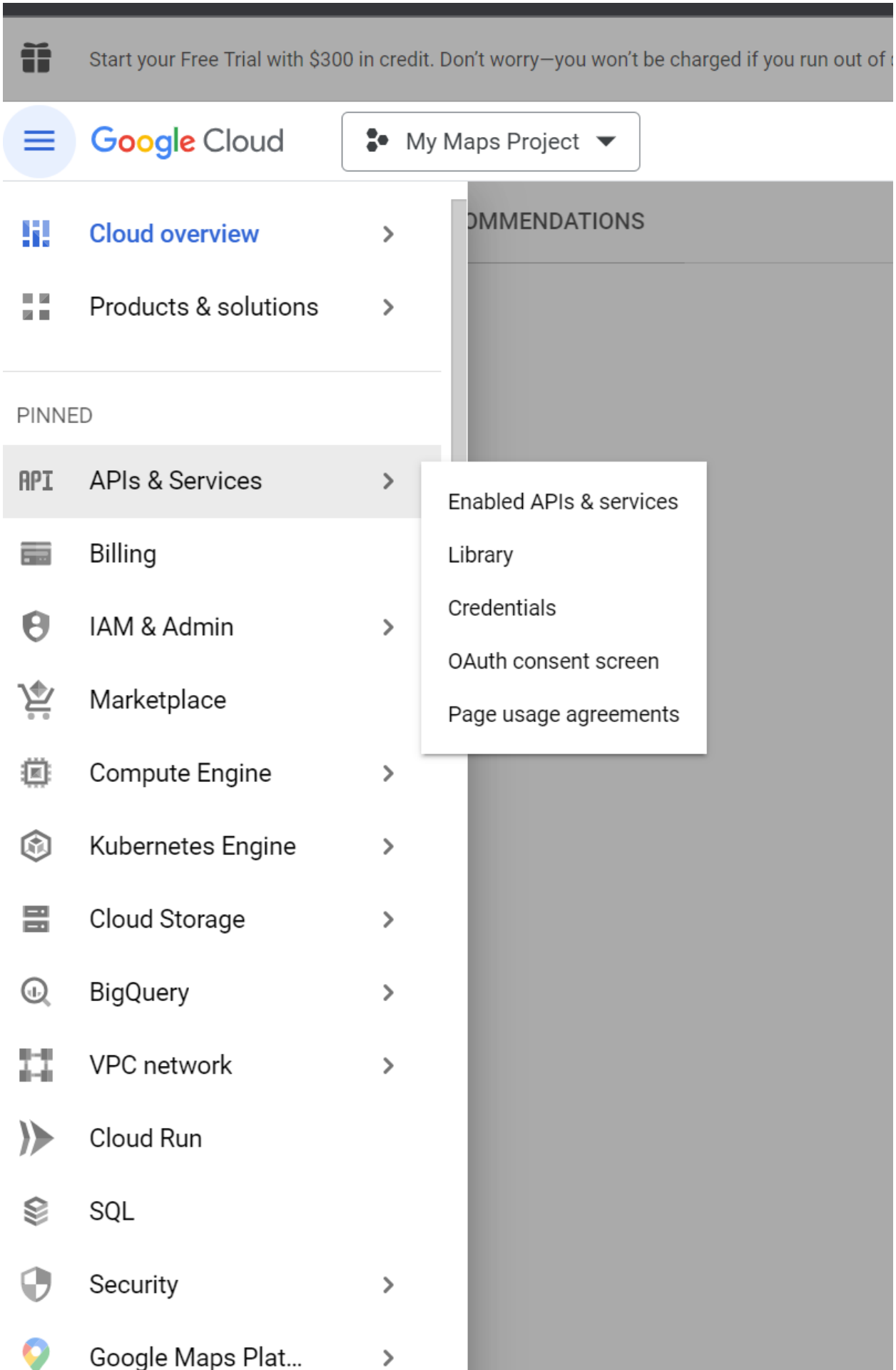




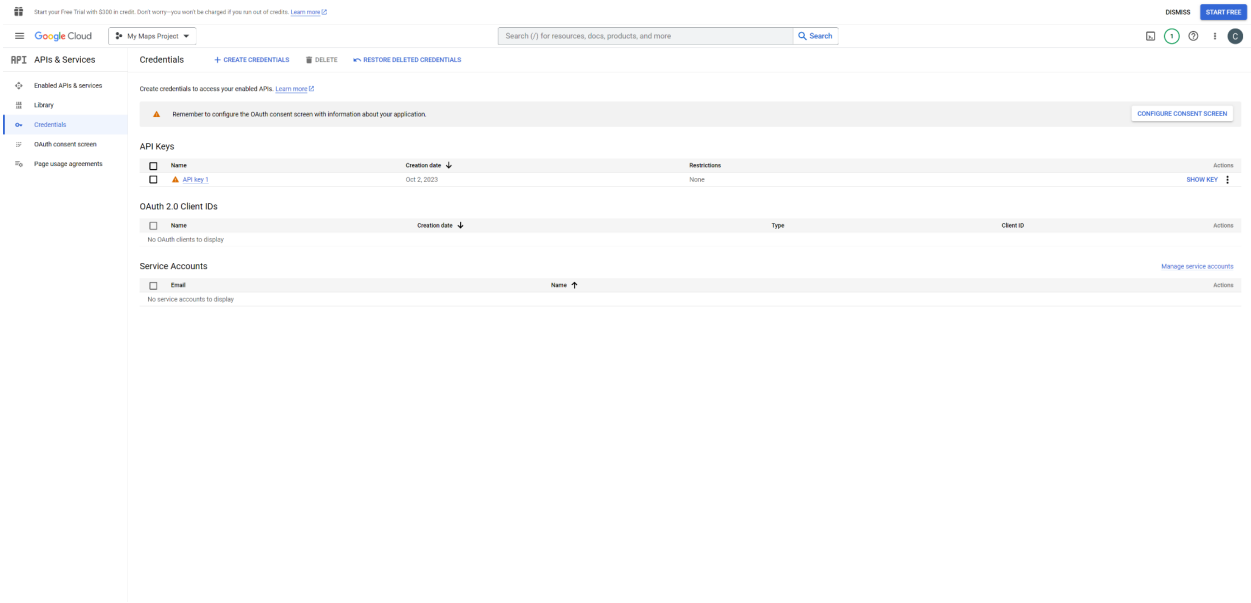
After registering and completing some survey questions, you'll get shown a screen that will have your API key information



API Credential Page:
Navigation Menu > APIs & Services > Credentials

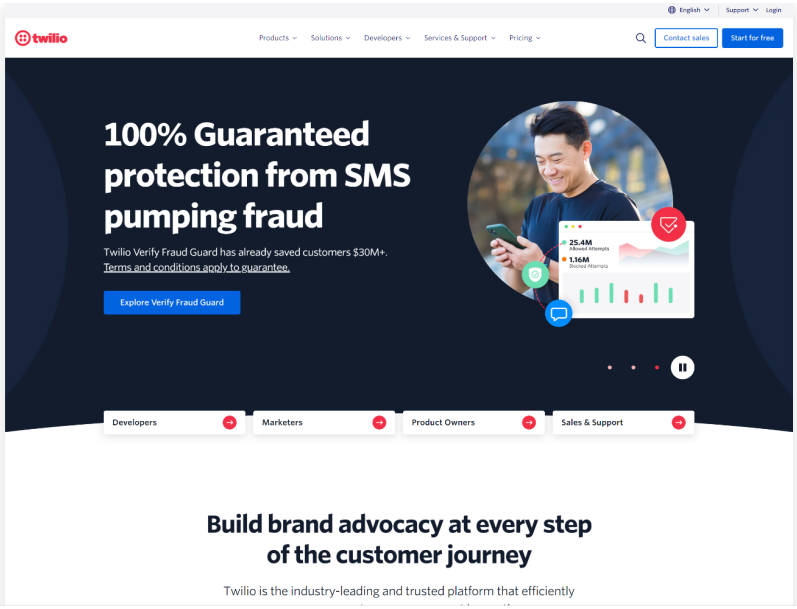

Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of
Google Cloud
My Maps Project
Cloud overview
Products & solutions
PINNED
APIs & Services
Billing
IAM & Admin
Marketplace
Compute Engine
Kubernetes Engine
Cloud Storage
BigQuery
VPC network
Cloud Run
SQL
Security
Google Maps Plat...
OMMENDATIONS
Enabled APIs & services
Library
Credentials
OAuth consent screen
Page usage agreements

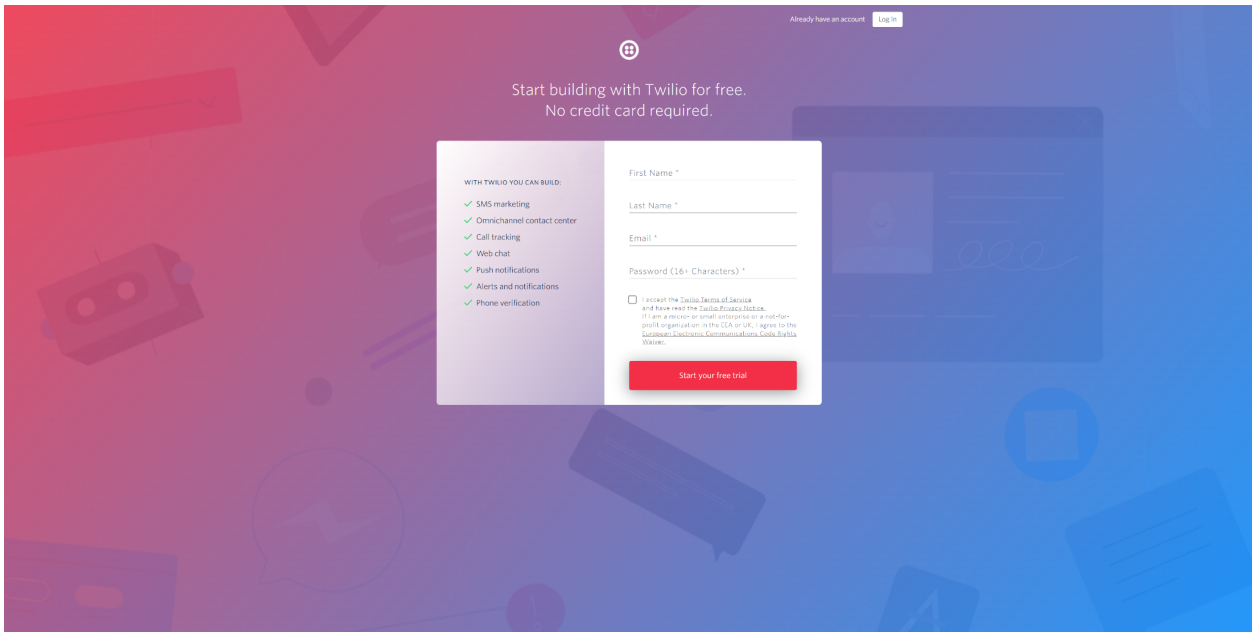
From the Credentials page, you can view or create a new API key if needed



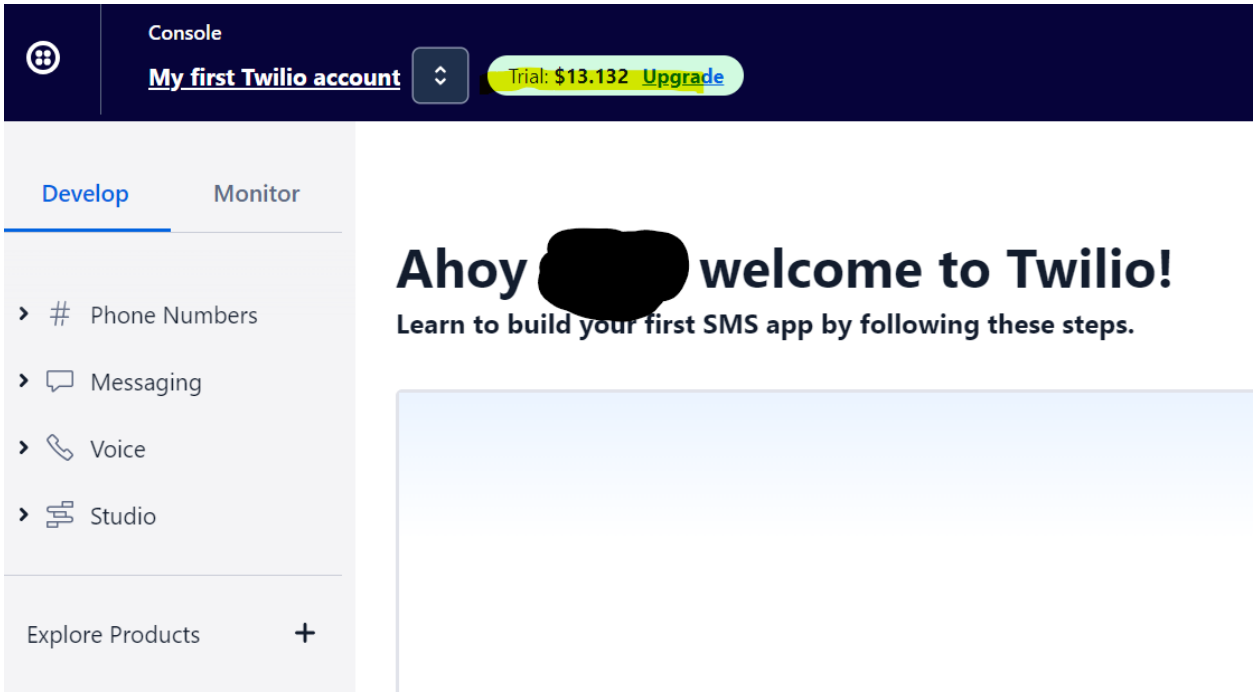
How to set up Twilio API key

Visit https://twilio.com, click Start for Free, and complete registration





Be aware that you will be given a trial credit of $15.00 for your API testing. You can view your balance in the top left of your console homepage



Follow the prompts to set up API-related information, such as your Account SID, Auth Token, and your Twilio phone number