



## ✓ Launch Sites Locations Analysis with Folium

Estimated time needed: **40** minutes

The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analyzing the existing launch site locations.

In the previous exploratory data analysis labs, you have visualized the SpaceX launch dataset using `matplotlib` and `seaborn` and discovered some preliminary correlations between the launch site and success rates. In this lab, you will be performing more interactive visual analytics using `Folium`.

## ✓ Objectives

This lab contains the following tasks:

- **TASK 1:** Mark all launch sites on a map
- **TASK 2:** Mark the success/failed launches for each site on the map
- **TASK 3:** Calculate the distances between a launch site to its proximities

After completed the above tasks, you should be able to find some geographical patterns about launch sites.

Let's first import required Python packages for this lab:

```
!pip3 install folium
!pip3 install wget
!pip3 install pandas
```



```
Requirement already satisfied: folium in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: branca>=0.6.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.11/dist-pac
```

```

Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: xyzservices in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/d
Collecting wget
  Downloading wget-3.2.zip (10 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: wget
  Building wheel for wget (setup.py) ... done
  Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9655 sha256=69
  Stored in directory: /root/.cache/pip/wheels/40/b3/0f/a40dbd1c6861731779f62cc4
Successfully built wget
Installing collected packages: wget
Successfully installed wget-3.2
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packag

```

```

import folium
import wget
import pandas as pd

```

```

# Import folium MarkerCluster plugin
from folium.plugins import MarkerCluster
# Import folium MousePosition plugin
from folium.plugins import MousePosition
# Import folium DivIcon plugin
from folium.features import DivIcon

```

If you need to refresh your memory about folium, you may download and refer to this previous folium lab:

[Generating Maps with Python](#)

## ✓ Task 1: Mark all launch sites on a map

First, let's try to add each site's location on a map using site's latitude and longitude coordinates

The following dataset with the name `spacex_launch_geo.csv` is an augmented dataset with latitude and longitude added for each site.

```
# Download and read the `spacex_launch_geo.csv`
spacex_csv_file = wget.download('https://cf-courses-data.s3.us.cloud-object-storage.
spacex_df=pd.read_csv(spacex_csv_file)
```

Now, you can take a look at what are the coordinates for each site.

```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `c
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

	Launch Site	Lat	Long	
0	CCAFS LC-40	28.562302	-80.577356	
1	CCAFS SLC-40	28.563197	-80.576820	
2	KSC LC-39A	28.573255	-80.646895	
3	VAFB SLC-4E	34.632834	-120.610745	

Next  
steps:

[Generate code with launch\\_sites\\_df](#)
[View recommended plots](#)
[New interactive sheet](#)

Above coordinates are just plain numbers that can not give you any intuitive insights about where are those launch sites. If you are very good at geography, you can interpret those numbers directly in your mind. If not, that's fine too. Let's visualize those locations by pinning them on a map.

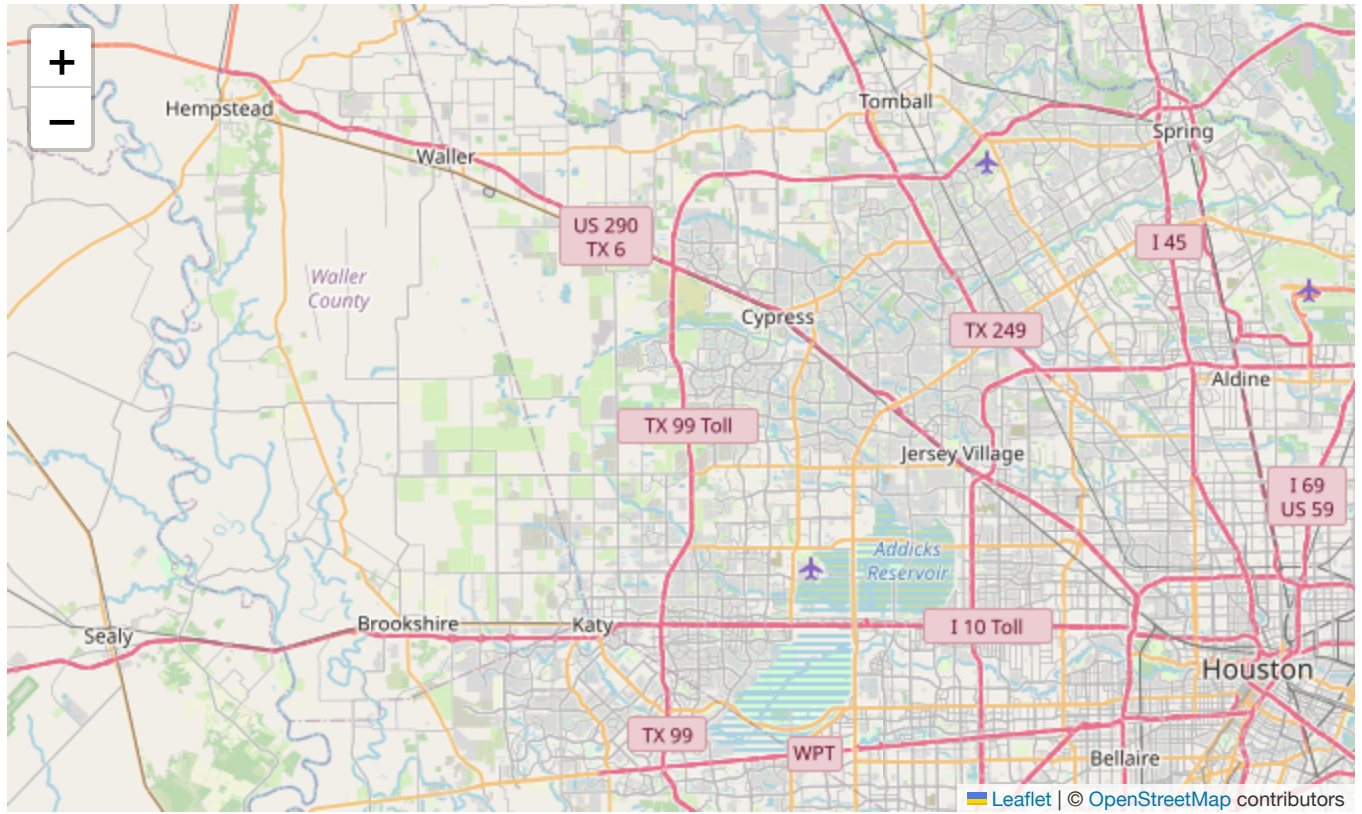
We first need to create a folium Map object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.

```
# Start location is NASA Johnson Space Center
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
```

We could use `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate. For example,

```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add
```

```
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JS
    )
)
site_map.add_child(circle)
site_map.add_child(marker)
```



and you should find a small yellow circle near the city of Houston and you can zoom-in to see a larger circle.

Now, let's add a circle for each launch site in data frame `launch_sites`

*TODO:* Create and add `folium.Circle` and `folium.Marker` for each launch site on the site map

An example of `folium.Circle`:

```
folium.Circle(coordinate, radius=1000, color='#000000',
fill=True).add_child(folium.Popup(...))
```

An example of `folium.Marker`:

```
folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0),
html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'label', ))
```

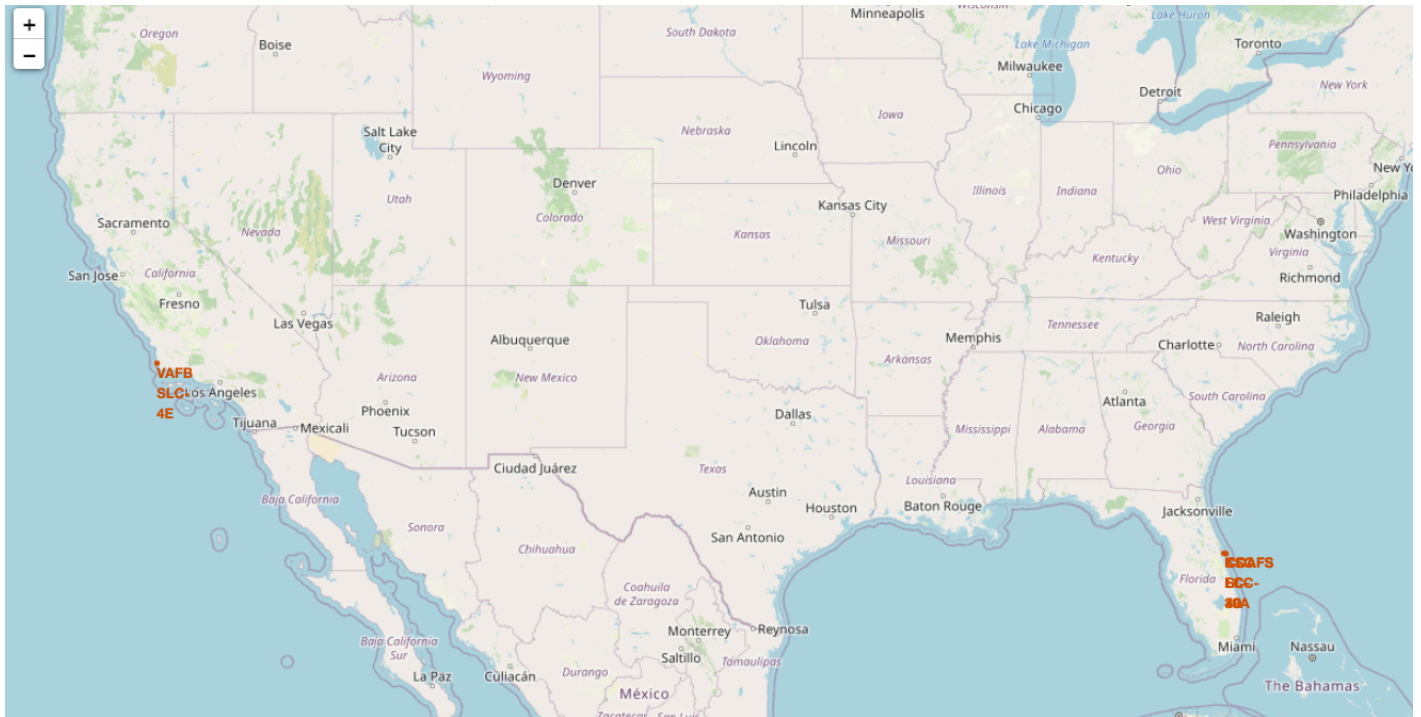
```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) valu

for index, row in launch_sites_df.iterrows():
    coordinate = [row['Lat'], row['Long']]
    launch_site_name = row['Launch Site']
    circle = folium.Circle(coordinate, radius=1000, color='#d35400', fill=True).add_
    # Create a blue circle at the row i with a icon showing its name
    marker = folium.map.Marker(
        coordinate,
        # Create an icon as a text label
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % laun
        )
    )
    site_map.add_child(circle)
    site_map.add_child(marker)

site_map
```



The generated map with marked launch sites should look similar to the following:



Now, you can explore the map by zoom-in/out the marked areas , and try to answer the following questions:

- Are all launch sites in proximity to the Equator line?
- Are all launch sites in very close proximity to the coast?

Also please try to explain your findings.



## Task 2: Mark the success/failed launches for each site on the map

Next, let's try to enhance the map by adding the launch outcomes for each site, and see which sites have high success rates. Recall that data frame `spacex_df` has detailed launch records, and the `class` column indicates if this launch was successful or not

```
spacex_df.tail(10)
```





	Launch Site	Lat	Long	class	
46	KSC LC-39A	28.573255	-80.646895	1	
47	KSC LC-39A	28.573255	-80.646895	1	
48	KSC LC-39A	28.573255	-80.646895	1	
49	CCAFS SLC-40	28.563197	-80.576820	1	
50	CCAFS SLC-40	28.563197	-80.576820	1	
51	CCAFS SLC-40	28.563197	-80.576820	0	
52	CCAFS SLC-40	28.563197	-80.576820	0	
53	CCAFS SLC-40	28.563197	-80.576820	0	
54	CCAFS SLC-40	28.563197	-80.576820	1	
55	CCAFS SLC-40	28.563197	-80.576820	0	

Next, let's create markers for all launch records. If a launch was successful (`class=1`), then we use a green marker and if a launch was failed, we use a red marker (`class=0`)

Note that a launch only happens in one of the four launch sites, which means many launch records will have the exact same coordinate. Marker clusters can be a good way to simplify a map containing many markers having the same coordinate.

Let's first create a `MarkerCluster` object

```
marker_cluster = MarkerCluster()
```


*TODO:* Create a new column in `launch_sites` dataframe called `marker_color` to store the marker colors based on the `class` value



```
# Apply a function to check the value of `class` column
# If class=1, marker_color value will be green
# If class=0, marker_color value will be red
```

```
# Function to assign color to launch outcome
def assign_marker_color(launch_outcome):
    if launch_outcome == 1:
        return 'green'
```

```
else:
    return 'red'
```

```
spacex_df['marker_color'] = spacex_df['class'].apply(assign_marker_color)
spacex_df.tail(10)
```



	Launch Site	Lat	Long	class	marker_color	
46	KSC LC-39A	28.573255	-80.646895	1	green	
47	KSC LC-39A	28.573255	-80.646895	1	green	
48	KSC LC-39A	28.573255	-80.646895	1	green	
49	CCAFS SLC-40	28.563197	-80.576820	1	green	
50	CCAFS SLC-40	28.563197	-80.576820	1	green	
51	CCAFS SLC-40	28.563197	-80.576820	0	red	
52	CCAFS SLC-40	28.563197	-80.576820	0	red	
53	CCAFS SLC-40	28.563197	-80.576820	0	red	
54	CCAFS SLC-40	28.563197	-80.576820	1	green	
55	CCAFS SLC-40	28.563197	-80.576820	0	red	

*TODO:* For each launch result in `spacex_df` data frame, add a `folium.Marker` to `marker_cluster`

```
# Add marker_cluster to current site_map
# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was succeeded
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color'])
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map

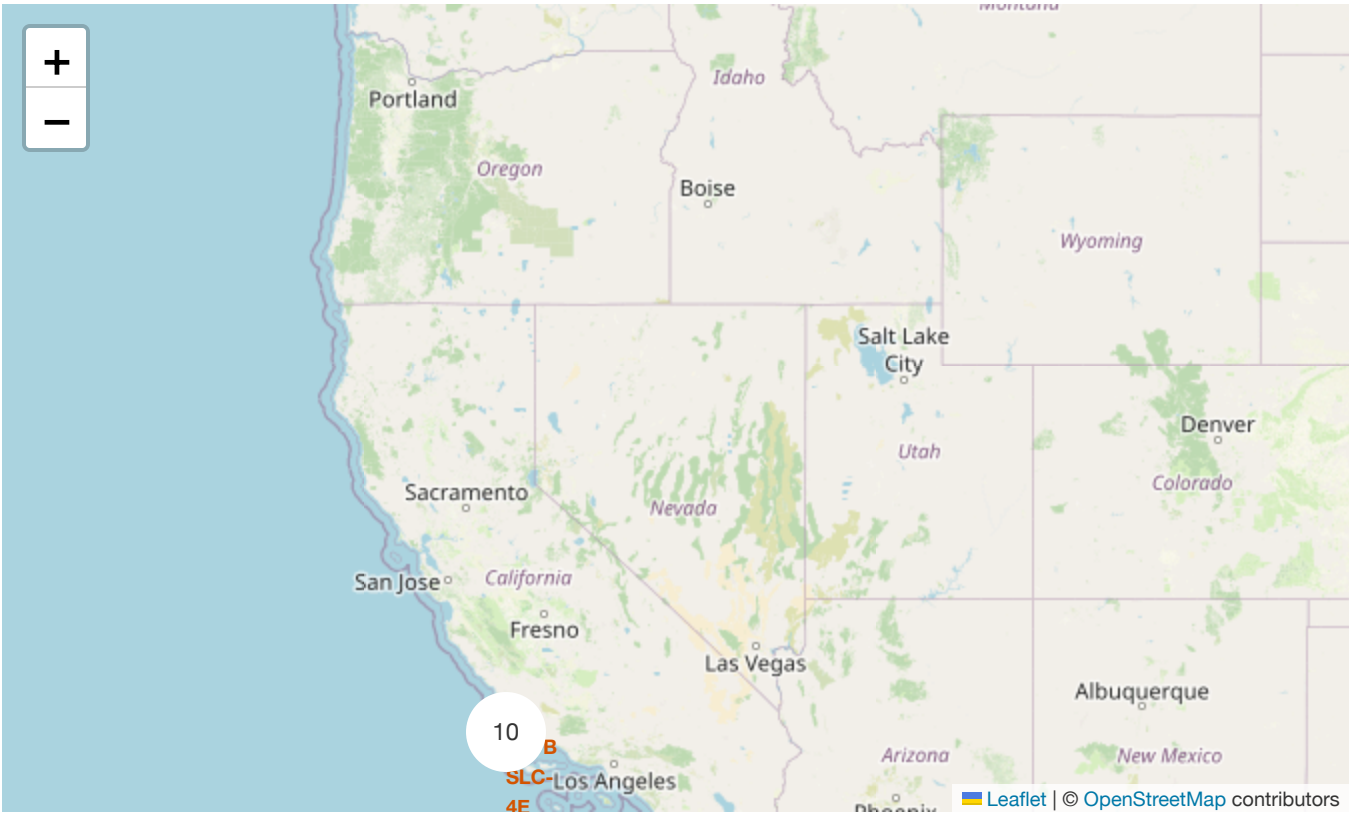
    coordinate = [record['Lat'], record['Long']]
    launch_site_name = record['Launch Site']
    icon_color = record['marker_color']

    marker = folium.map.Marker(
        coordinate,
        # Create an icon as a text label
        icon=folium.Icon(color='white', icon_color=icon_color)
    )

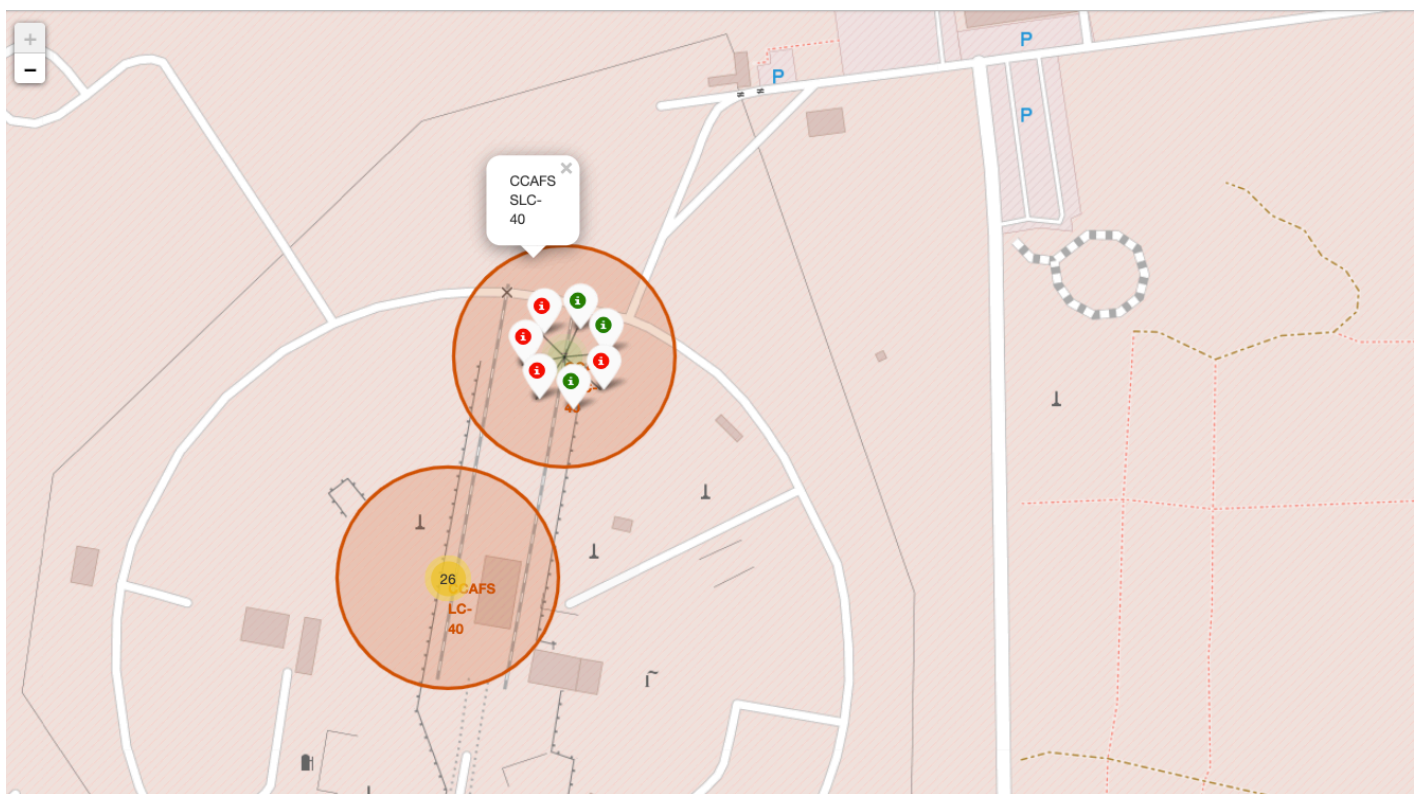
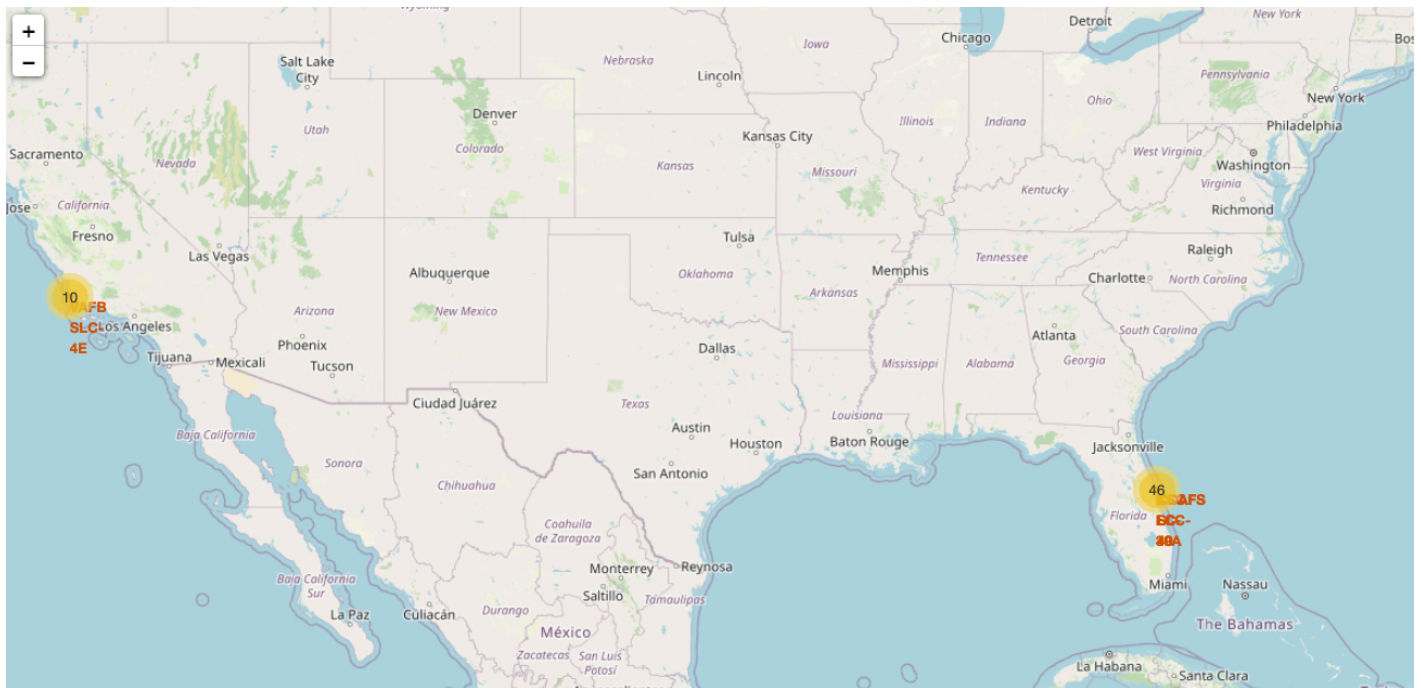
    marker_cluster.add_child(marker)

site_map.add_child(marker_cluster)

site_map
```



Your updated map may look like the following screenshots:



From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.

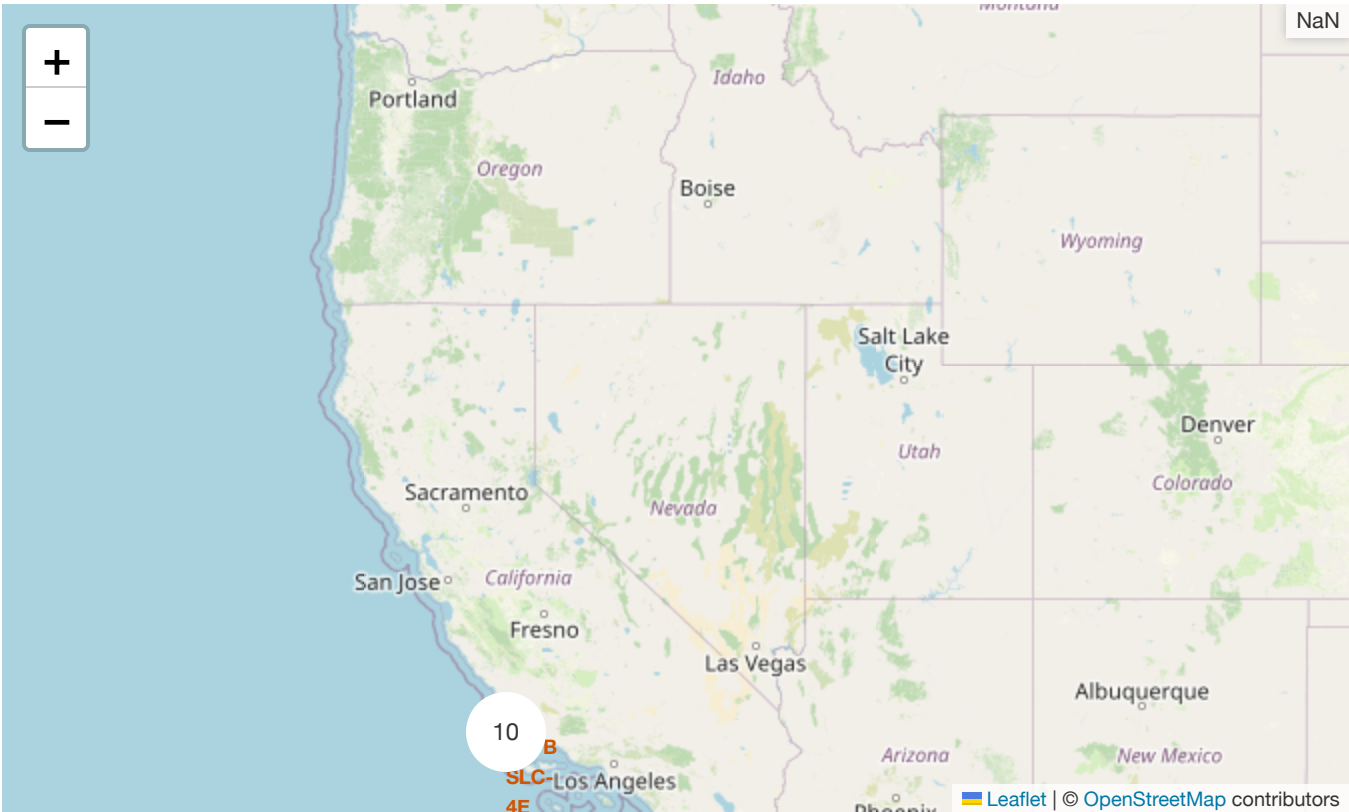
## ✓ TASK 3: Calculate the distances between a launch site to its proximities

Next, we need to explore and analyze the proximities of launch sites.

Let's first add a `MousePosition` on the map to get coordinate for a mouse over a point on the map. As such, while you are exploring the map, you can easily find the coordinates of any points of interests (such as railway)

```
# Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map
formatter = "function(num) {return L.Util.formatNum(num, 5)};"
mouse_position = MousePosition(
    position='topright',
    separator=' Long: ',
    empty_string='NaN',
    lng_first=False,
    num_digits=20,
    prefix='Lat:',
    lat_formatter=formatter,
    lng_formatter=formatter,
)

site_map.add_child(mouse_position)
site_map
```



Now zoom in to a launch site and explore its proximity to see if you can easily find any railway, highway, coastline, etc. Move your mouse to these points and mark down their coordinates (shown on the top-left) in order to the distance to the launch site.

You can calculate the distance between two points on the map based on their Lat and Long values using the following method:

```
from math import sin, cos, sqrt, atan2, radians

def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

*TODO:* Mark down a point on the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site.

```
# find coordinate of the closet coastline

coastline_lat = 28.56367
coastline_lon = -80.57163

spacex_df['distance_coastline'] = spacex_df.apply(lambda row: calculate_distance(row
spacex_df.tail()
```





	Launch Site	Lat	Long	class	marker_color	distance_coastline
51	CCAFS SLC-40	28.563197	-80.57682	0	red	0.509744
52	CCAFS SLC-40	28.563197	-80.57682	0	red	0.509744
53	CCAFS SLC-40	28.563197	-80.57682	0	red	0.509744



*TODO:* After obtained its coordinate, create a `folium.Marker` to show the distance

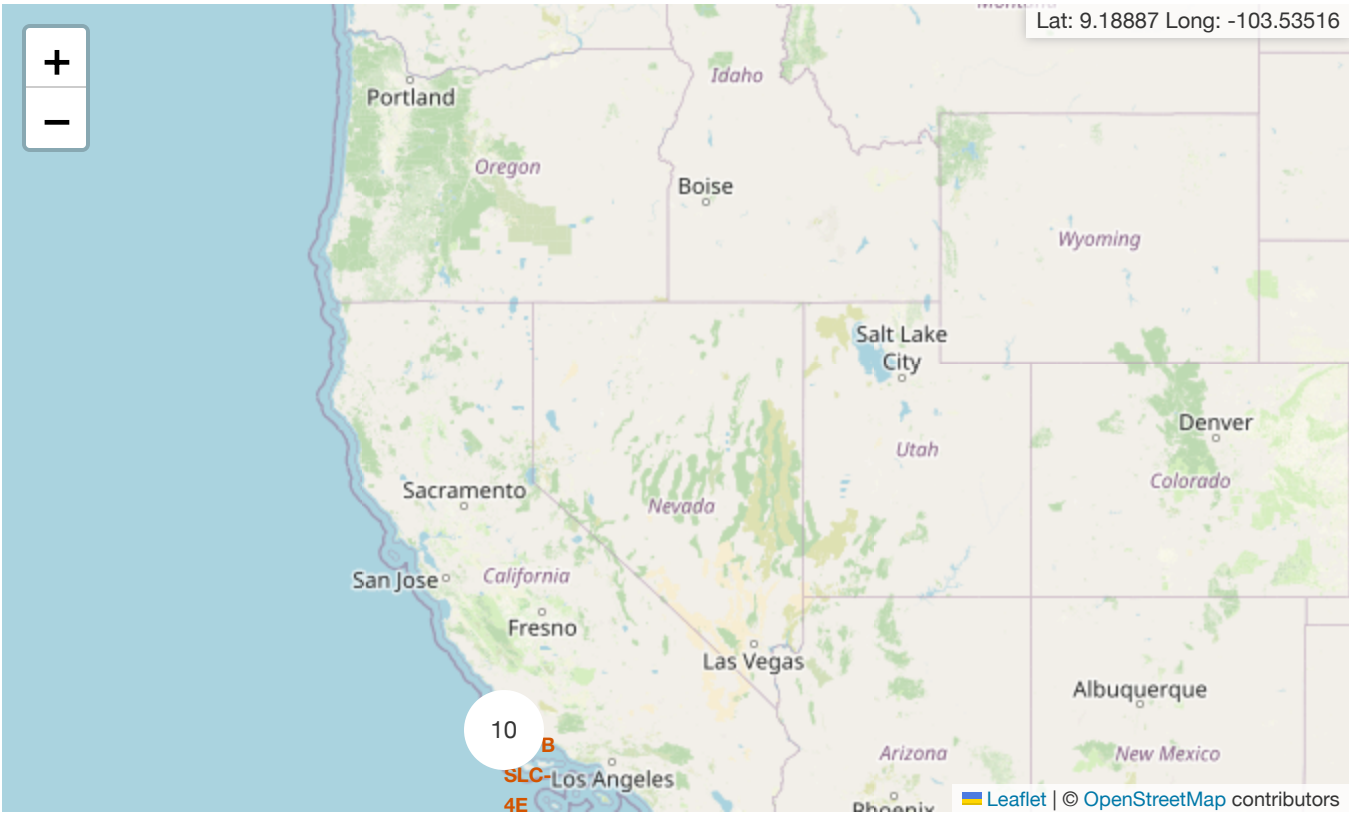
```
# Create and add a folium.Marker on your selected closest coastline point on the map
# Display the distance between coastline point and launch site using the icon proper
# for example
```

```
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map

    coordinate = [record['Lat'], record['Long']]
    launch_site_name = record['Launch Site']
    icon_color = record['marker_color']
    distance = record['distance_coastline']

    distance_marker = folium.Marker(
        coordinate,
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b>'
        )
    )

    site_map.add_child(distance_marker)
```



*TODO:* Draw a PolyLine between a launch site to the selected coastline point

```
# Create a `folium.PolyLine` object using the coastline coordinates and launch site
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map

    coordinates = [[record['Lat'], record['Long']], [coastline_lat, coastline_lon]]

    lines=folium.PolyLine(locations=coordinates, weight=1)

    site_map.add_child(lines)

site_map
```



Lat: 16.23577 Long: 2.04346