Task 1: Data Collection Agent

To set up a real satellite API with Crew.ai, the process involves a few key steps. The goal is to configure your Crew.ai agent to communicate with an actual satellite data provider API, fetch telemetry data, and handle the data as needed. Here's a detailed guide to help you integrate a real satellite API:

1. Choose a Satellite Data Provider

There are several satellite data providers that offer APIs to access telemetry, imagery, and other forms of satellite data. Some of the common providers include:

- **NASA's Earth Observation System Data and Information System (EOSDIS)**: Provides access to earth science data.
- **Sentinel Hub**: Offers Sentinel satellite data (from ESA's Sentinel missions).
- **OpenWeatherMap**: Offers satellite-based weather data.
- **Planet Labs**: Provides high-resolution satellite imagery data.
- **NOAA**: Provides access to NOAA's satellite data.

Each provider will have different types of data, such as telemetry data, imagery, or other remote sensing information. Choose one based on your needs.

2. Obtain API Access

Once you've chosen a satellite data provider, you need to sign up for an API key (if required). Most services require you to create an account and generate an API key or authentication token to make requests.

For example, if you're using **Sentinel Hub**, you would:

1. Go to the Sentinel Hub website.
2. Sign up for an account.
3. Navigate to the dashboard and generate an API key.

For other providers, follow a similar process to get the API credentials.

3. Understand the API Documentation

Read the documentation of the satellite data provider to understand how to interact with the API. This will include details about:

- **Endpoints**: The URL paths where you can request satellite data.
- **Authentication**: How to pass your API key or access token in requests.
- **Data Format**: The format of the data being returned (JSON, GeoTIFF, etc.).
- **Parameters**: How to specify what data you want (e.g., coordinates, date range, sensor type).

Here's an example API request using **NASA's Earthdata API**:

```
GET
https://api.nasa.gov/planetary/earth/assets?lon={longitude}&lat={latitude}&dim
={dimension}&date={date}&api_key={YOUR_API_KEY}
```

4. Setup Crew.ai Agent

Within Crew.ai, configure your **Data Fetcher** agent to interact with the satellite API. You'll need to input your API endpoint and credentials into Crew.ai's agent configuration.

Example Agent Setup in Crew.ai:

- **Role**: Data Fetcher
- **Goal**: Fetch satellite telemetry data
- **API Endpoint**: https://api.nasa.gov/planetary/earth/assets
- **Authentication**: Provide your API key in the request headers or as a URL parameter.
- **Tools**: Make sure the agent has access to the tools required for making HTTP requests (like requests or API Access).

5. Write the Fetch Logic in Crew.ai

Depending on Crew.ai structures agents, you can write or configure a function within the agent to fetch data from the real satellite API.

Here's an example (conceptual) code block for fetching data from a real satellite API in Crew.ai (Python-like pseudocode):

```
import crewai
import requests


class SatelliteDataFetcher(crewai.Agent):
    def __init__(self, api_key):
        self.api_key = api_key


    def fetch_satellite_data(self, lat, lon, date):
        url = f"https://api.nasa.gov/planetary/earth/assets"
        params = {
            'lon': lon,
            'lat': lat,
            'date': date,
            'dim': 0.1,  # dimension of the area
            'api_key': self.api_key
        }
    response = requests.get(url, params=params)


    if response.status_code == 200:
      data = response.json()
      return data
    else:
      print(f"Failed to fetch data: {response.status_code}")
      return None


# Example usage
api_key = "your_nasa_api_key"
fetcher = SatelliteDataFetcher(api_key)
data = fetcher.fetch_satellite_data(lat=34.0522, lon=-118.2437, date='2024-01-01')


if data:
  print("Satellite data fetched successfully!")
  print(data)
else:
  print("Failed to retrieve data.")
```

6. Automate Data Fetching in Crew.ai

Crew.ai may allow you to automate data fetching tasks based on a schedule or trigger. Configure your agent to automatically query the satellite API at regular intervals or when new data is available.

7. Handling the Data

Once the data is fetched, you can either:

- **Pre-process the data**: Perform operations like filtering or cleaning the raw telemetry or imagery data.
- **Relay it for analysis**: Forward the data to other Crew.ai agents or systems for further analysis.

8. Example for Sentinel Hub API (Using Real Data)

```
import requests
url = 'https://services.sentinel-hub.com/api/v1/process'
headers = {
  'Authorization': 'Bearer your_sentinel_hub_token'
}
```

```
payload = {
    'input': {
        'bounds': {
            'bbox':       [13.822174072265625,       45.85080395917834,       14.55963134765625,
46.29191774991382]
        },
        'data': [
            {'type': 'sentinel-2-l1c'}
        ]
    },
    'output': {
        'width': 512,
        'height': 512,
        'responses': [
            {'identifier': 'default', 'format': {'type': 'image/tiff'}}
        ]
    }
}


response = requests.post(url, json=payload, headers=headers)


if response.status_code == 200:
    with open('satellite_image.tiff', 'wb') as f:
        f.write(response.content)
    print("Satellite data fetched successfully and saved as TIFF image.")
else:
    print(f"Error fetching satellite data: {response.status_code}")
```

Final Steps:

- **API Integration**: Make sure the API key and endpoint details are correct for your chosen provider.
- **Agent Configuration**: Set up the agent properly in Crew.ai’s interface to make periodic or on-demand requests.
- **Data Handling**: Decide how to process and store the fetched data once received.