MultiOn: Twitter News Agent

View on GitHub

In this example, we use Modal to deploy a cron job that periodically checks for AI news everyday and tweets it on Twitter using the MultiOn Agent API.

Import and define the app

Let's start off with imports, and defining a Modal app.

```
import os

import modal

app = modal.App("multion-news-tweet-agent")
```

Searching for AI News

Let's also define an image that has the `multion` package installed, so we can query the API.

```
multion_image = modal.Image.debian_slim().pip_install("multion")
```

We can now define our main entrypoint, that uses MultiOn to scrape AI news everyday and post it on our twitter account. We specify a schedule in the function decorator, which means that our function will run automatically at the given interval.

Set up MultiOn

MultiOn is a next-gen Web Action Agent that can take actions on behalf of the user. You can watch it in action here: Youtube demo.

The MultiOn API enables building the next level of web automation & custom AI agents capable of performing complex actions on the internet with just a few lines of code.

To get started, first create an account with MultiOn, install the MultiOn chrome extension and login to your Twitter account in your browser. To use the API create a MultiOn API Key and store it as a modal secret on the dashboard

```
@app.function(
    image=multion_image, secrets=[modal.Secret.from_name("MULTION_API_KEY")]
)
def news_tweet_agent():
    # Import MultiOn
    import multion

    # Login to MultiOn using the API key
    multion.login(use_api=True, multion_api_key=os.environ["MULTION_API_KEY"])

    # Enable the Agent to run locally
    multion.set_remote(False)

    params = {
        "url": "https://www.multion.ai",
        "cmd": "Go to twitter (im already signed in). Search for the last tweets i made (c
        "maxSteps": 100,
    }

    response = multion.browse(params)

    print(f"MultiOn response: {response}")
```

Test running

We can now test run our scheduled function as follows: `modal run multion_news_agent.py.py::app.news_tweet_agent`

Defining the schedule and deploying

Let's define a function that will be called by Modal every day.

```
@app.function(schedule=modal.Cron("0 9 * * *"))
def run_daily():
    news_tweet_agent.remote()
```

In order to deploy this as a persistent cron job, you can run `modal deploy multion_news_agent.py`.

Once the job is deployed, visit the apps page page to see its execution history, logs and other stats.