

[Featured](#)[Getting started](#)[Hello, world](#)[Simple web scraper](#)[Serving web endpoints](#)[Large language models \(LLMs\)](#)

Hacker News Slackbot

[View on GitHub](#)

In this example, we use Modal to deploy a cron job that periodically queries Hacker News for new posts matching a given search term, and posts the results to Slack.

Import and define the app

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

```
import os
from datetime import datetime, timedelta

import modal

app = modal.App("example-hn-bot")
```

Now, let's define an image that has the `slack-sdk` package installed, in which we can run a function that posts a slack message.

```
slack_sdk_image = modal.Image.debian_slim().pip_install("slack-sdk")
```

Defining the function and importing the secret

Our Slack bot will need access to a bot token. We can use Modal's [Secrets](#) interface to accomplish this. To quickly create a Slack bot secret, navigate to the [create secret](#) page, select the Slack secret template from the list options, and follow the instructions in the "Where to find the credentials?" panel. Name your secret `hn-bot-slack`, so that the code in this example still works.

Now, we define the function `post_to_slack`, which simply instantiates the Slack client using our token, and then uses it to post a message to a given channel name.

```
@app.function(
    image=slack_sdk_image, secrets=[modal.Secret.from_name("hn-bot-slack")]
)
async def post_to_slack(message: str):
    import slack_sdk

    client = slack_sdk.WebClient(token=os.environ["SLACK_BOT_TOKEN"])
    client.chat_postMessage(channel="hn-alerts", text=message)
```

Searching Hacker News

We are going to use Algolia's [Hacker News Search API](#) to query for posts matching a given search term in the past X days. Let's define our search term and query period.

```
QUERY = "serverless"
WINDOW_SIZE_DAYS = 1
```

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

```
requests_image = modal.Image.debian_slim().pip_install("requests")
```

We can now define our main entrypoint, that queries Algolia for the term, and calls `post_to_slack` on all the results. We specify a [schedule](#) in the function decorator, which means that our function will run automatically at the given interval.

```
@app.function(image=requests_image)
def search_hackernews():
    import requests

    url = "http://hn.algolia.com/api/v1/search"

    threshold = datetime.utcnow() - timedelta(days=WINDOW_SIZE_DAYS)

    params = {
```

```
"query": QUERY,
"numericFilters": f"created_at_i>{threshold.timestamp()}",
}

response = requests.get(url, params, timeout=10).json()
urls = [item["url"] for item in response["hits"] if item.get("url")]

print(f"Query returned {len(urls)} items.")

post_to_slack.for_each(urls)
```

Test running

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

Defining the schedule and deploying

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

```
@app.function(schedule=modal.Period(days=1))
def run_daily():
    search_hackernews.remote()
```

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

Once the job is deployed, visit the [apps page](#) to see its execution history, logs and other stats.



© 2024

[About](#)

[Slack Community](#)

[Status](#)

[Pricing](#)

[Changelog](#)

[Examples](#)

[Documentation](#)