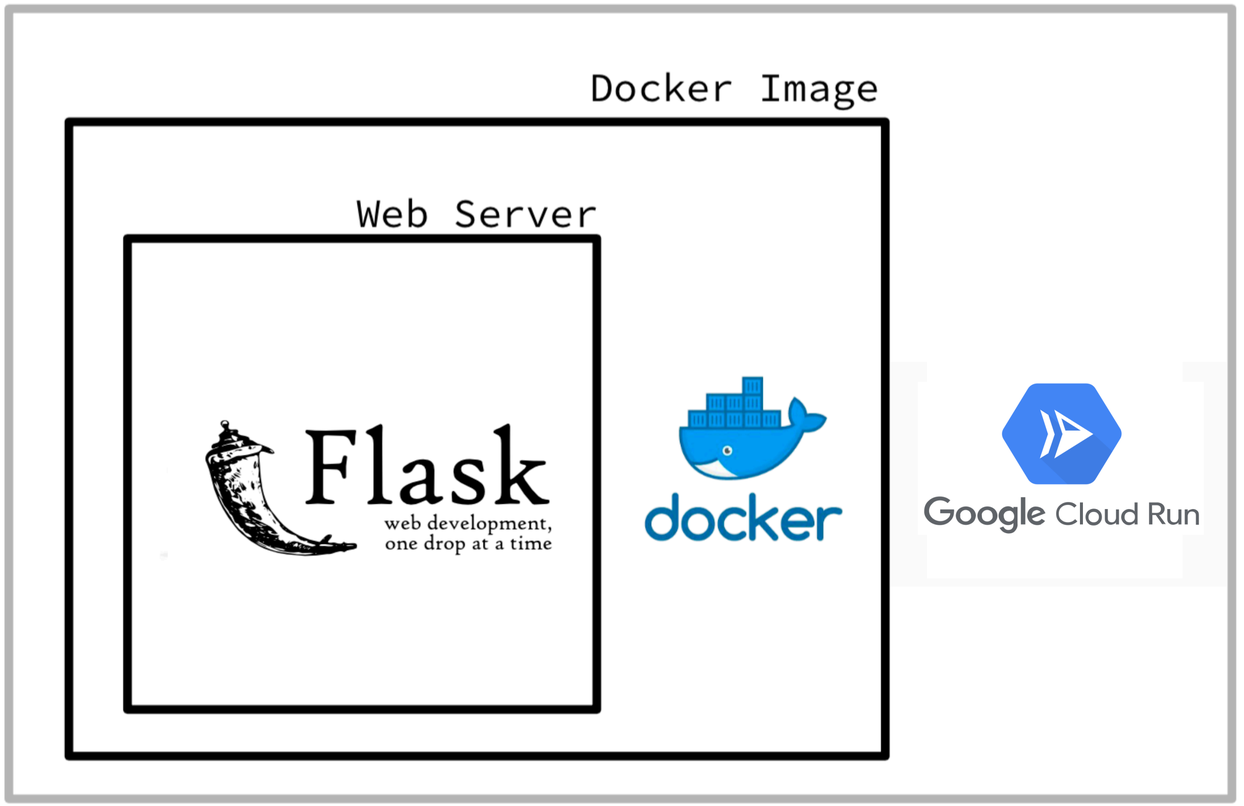
What are we doing here?

Let’s say we’ve created a machine learning model…now we want someone to use it!

Lets start to think about how we can deploy our model so someone else can access it (without knowing much of what’s going on underneath)

Expose a web API

What are we going to build?



What is an API anyways?

Technically, API stands for **Application Programming Interface**. At some point or another, most large companies have built APIs for their customers, or for internal use.

When you type www.facebook.com into your browser, a request goes out to Facebook's remote server. Once your browser receives the response, it interprets the code and displays the page.

An API isn't the same as the remote server—rather it is the part of the server that receives requests and sends responses.

When a company offers an API to their customers, it just means that they've built a set of dedicated URLs that return pure data responses—meaning the responses won't contain the kind of presentational overhead that you would expect in a graphical user interface like a website. (thanks https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82)

REST API

A REST API defines a set of functions which developers can perform requests and receive responses via HTTP protocol such as GET and POST.

REST stands for Representational state transfer which essentially refers to a style of web architecture that has many underlying characteristics and governs the behavior of clients and servers.

An API can be considered “RESTful” if it has the following features (main features of R):

Client–server – The client handles the front end the server handles the backend and can both be replaced independently of each other.

Stateless – No client data is stored on the server between requests and session state is stored on the client.

Cacheable – Clients can cache response (just like browsers caching static elements of a web page) to improve performance

What is Flask?

A microframework for Python, meaning it has little to no dependencies to external libraries.

It really is a web framework providing tools, libraries and technologies to build web applications. (in our case an API)

Flask can create a REST API that allows you to send data, and receive a prediction as a response.

What are we doing with Flask?

We will create routes for our api that receives different requests (GET and POST), specifically a POSTing json data

Once our flask server receives json data it will apply a function and then return some value also in json format

What are we doing with Flask?

Last week we created a locally deployed API to help us understand the process through which we could Our local API was a Flask app contained in a docker container…the portability of the docker container will come in handy here.

Now let’s move this idea to a cloud based serverless platform, a Google Cloud Run service instance (we get plenty of free resources!), where people or applications outside our local machines can access our API

It should be noted that this is still simply to give us an idea about the deployment process, but we will be still falling a bit short of full-on productionalization.

Cloud Run

Build apps or websites quickly on a fully managed platform

Run frontend and backend services, batch jobs, host LLMs, and queue processing workloads without the need to manage infrastructure.

Get two million requests free per month.

Try it in console | Contact sales

Product highlights

- The flexibility of containers with the simplicity of serverless
- Start with source code and run your app in 20+ regions at once
- Only pay when your code is running
- Deploy and build web apps and websites

Cloud Run in a

Google Cloud | Gemini API | Search (/) for resources, docs, products, and more | Search

Cloud Run | Services | Deploy container | Connect repo | Write a function | Release Notes

Services | Jobs

Service
Each service has a unique endpoint & autoscales deployed code.

Job
Execute code to completion.

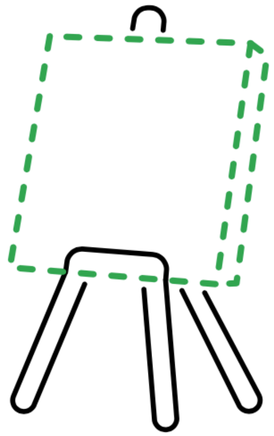
A service exposes a unique endpoint and autom...sts.
Deploy a container image, source code or a func

Services

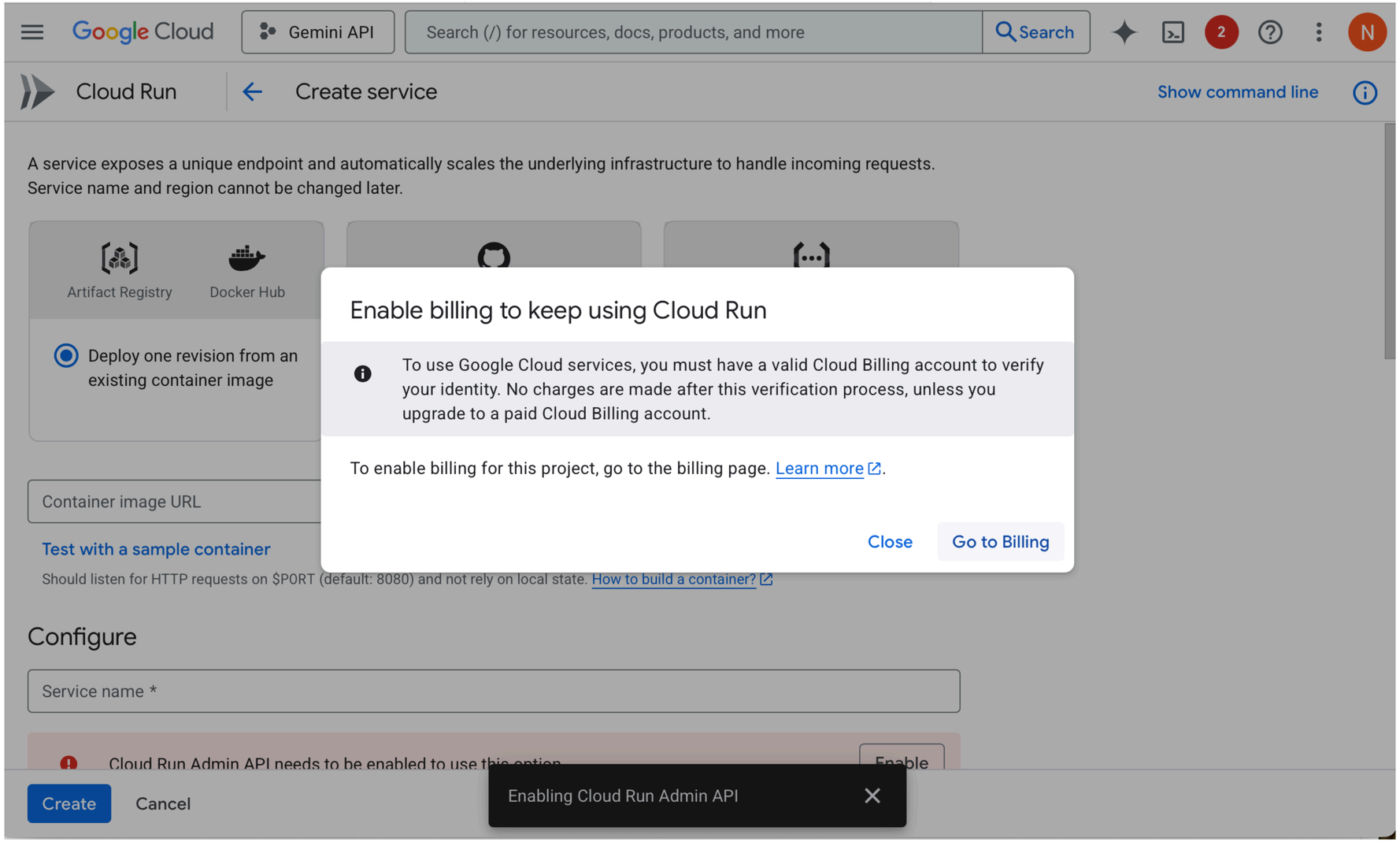
Filter | Filter services

Name	Deployment type	Req/sec	Region	Authentication	Ingress	Recommendation	Last deploy

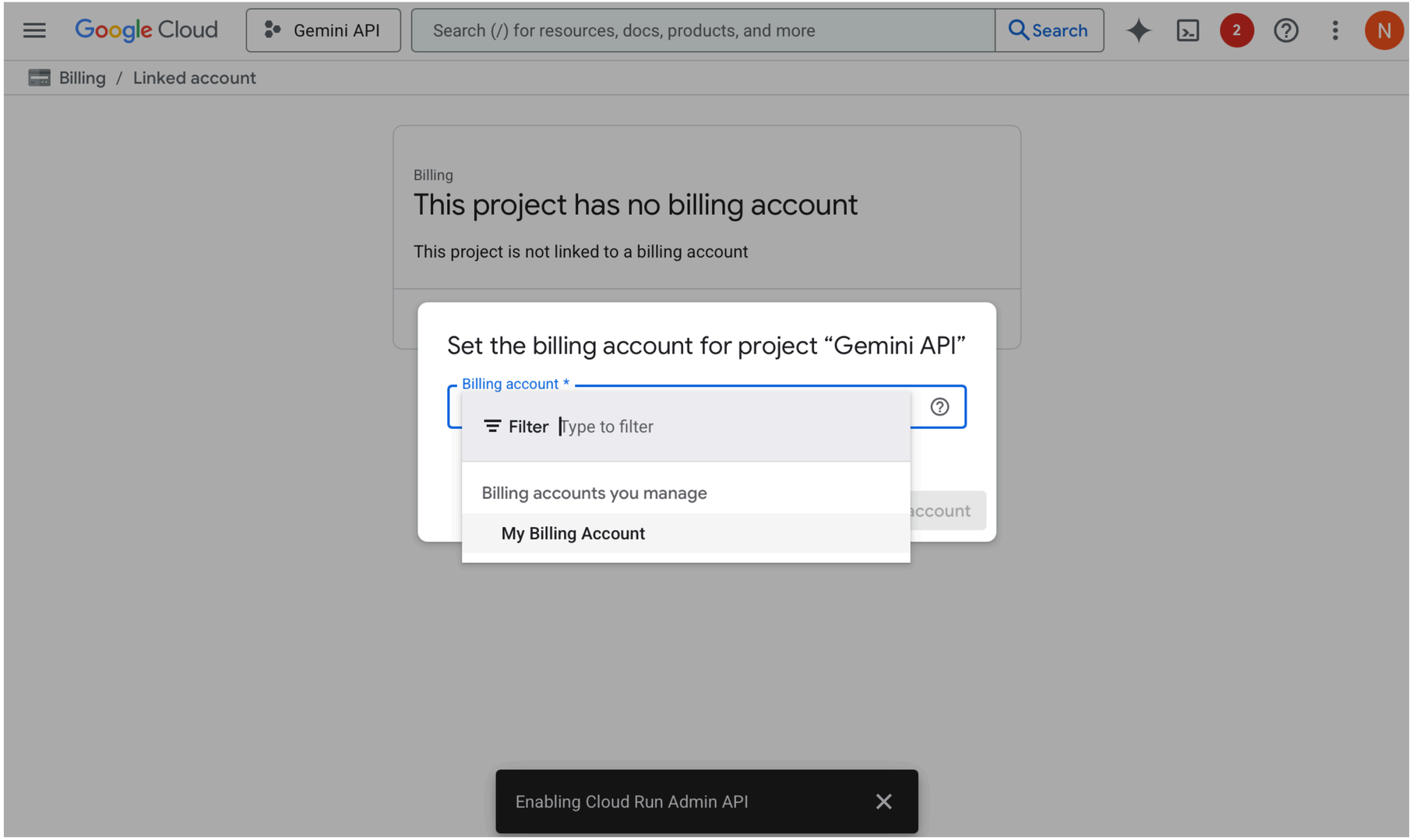
No results to display



You will need to enable billing to use

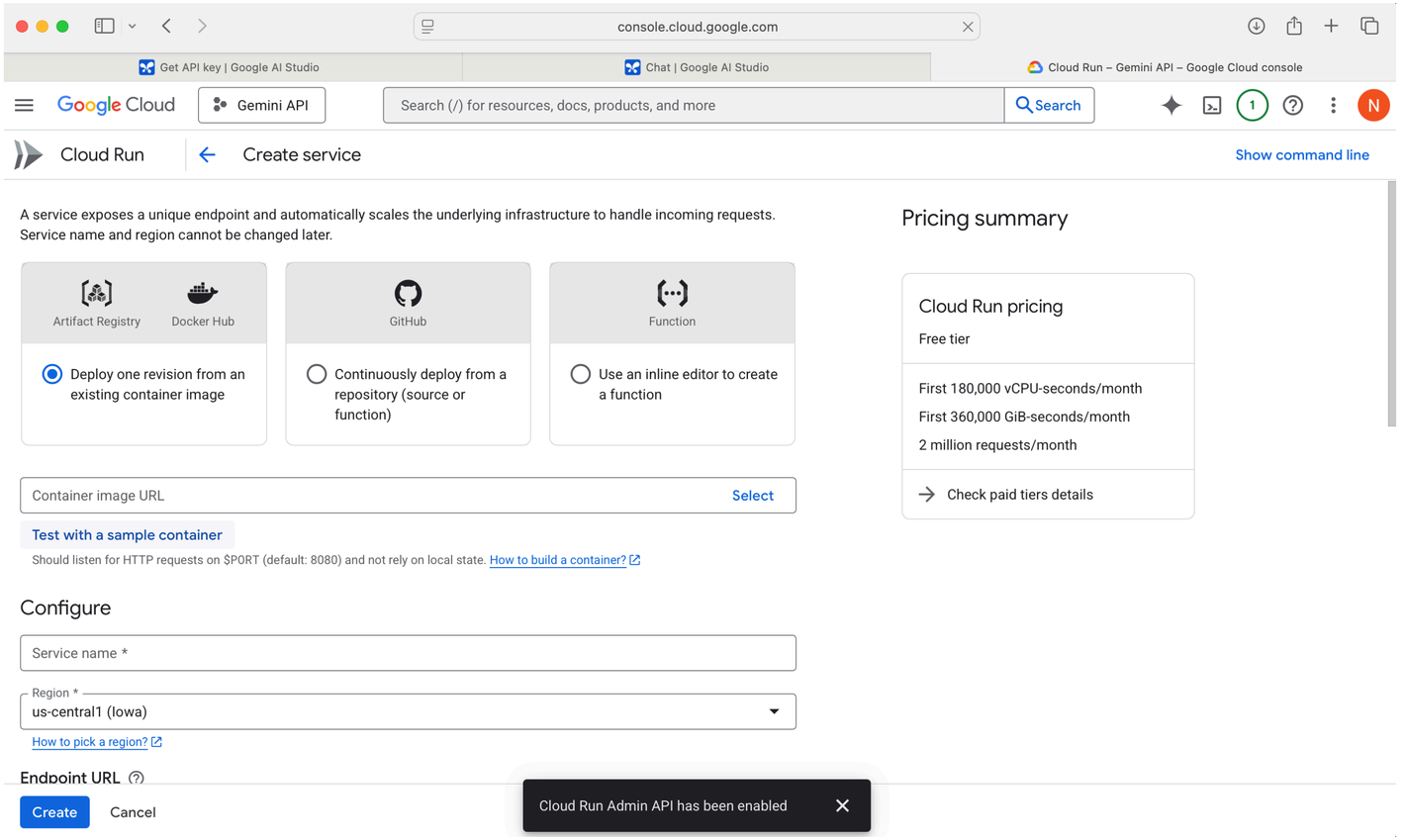


You will need to enable billing to use

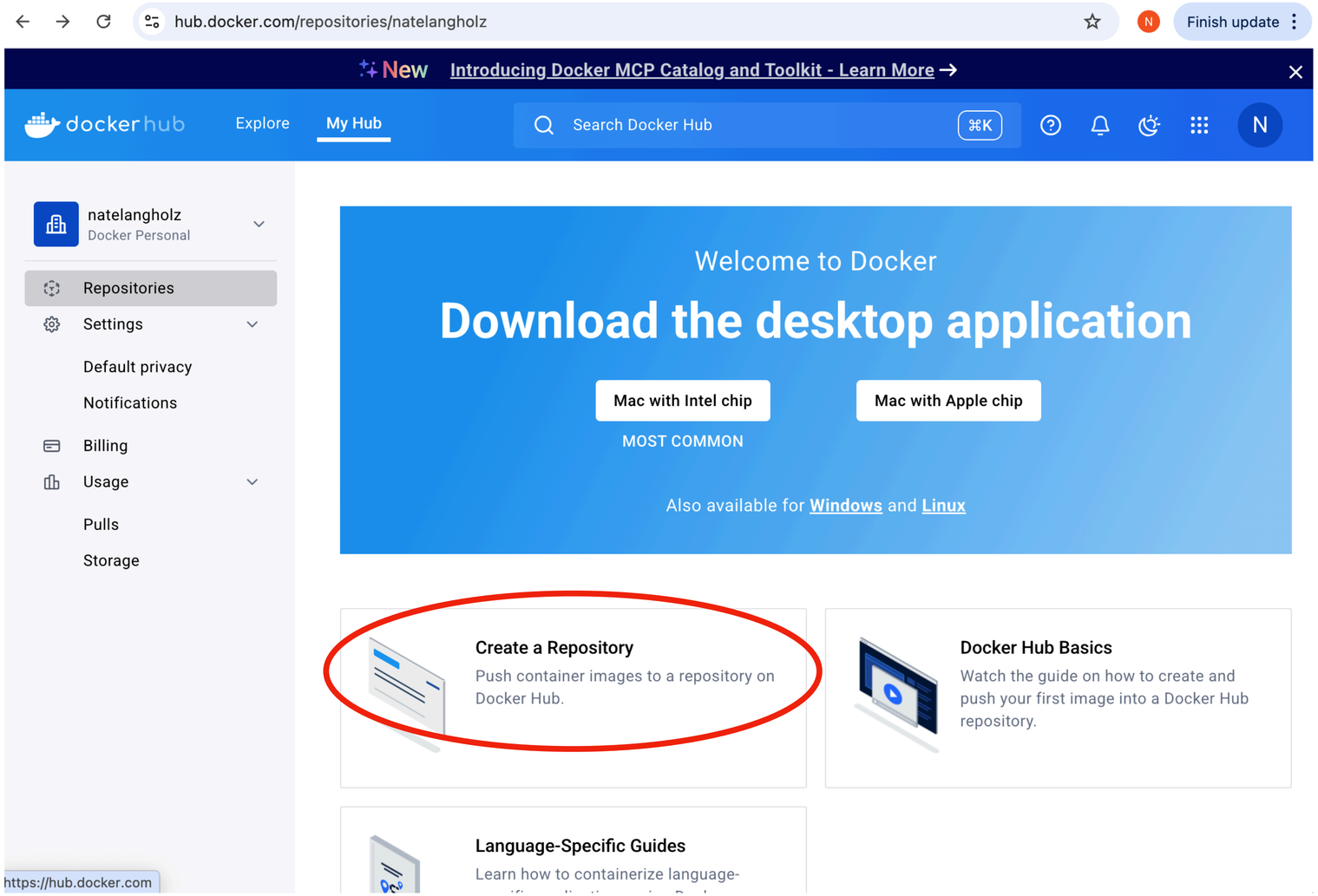


Once you have lets create a service.

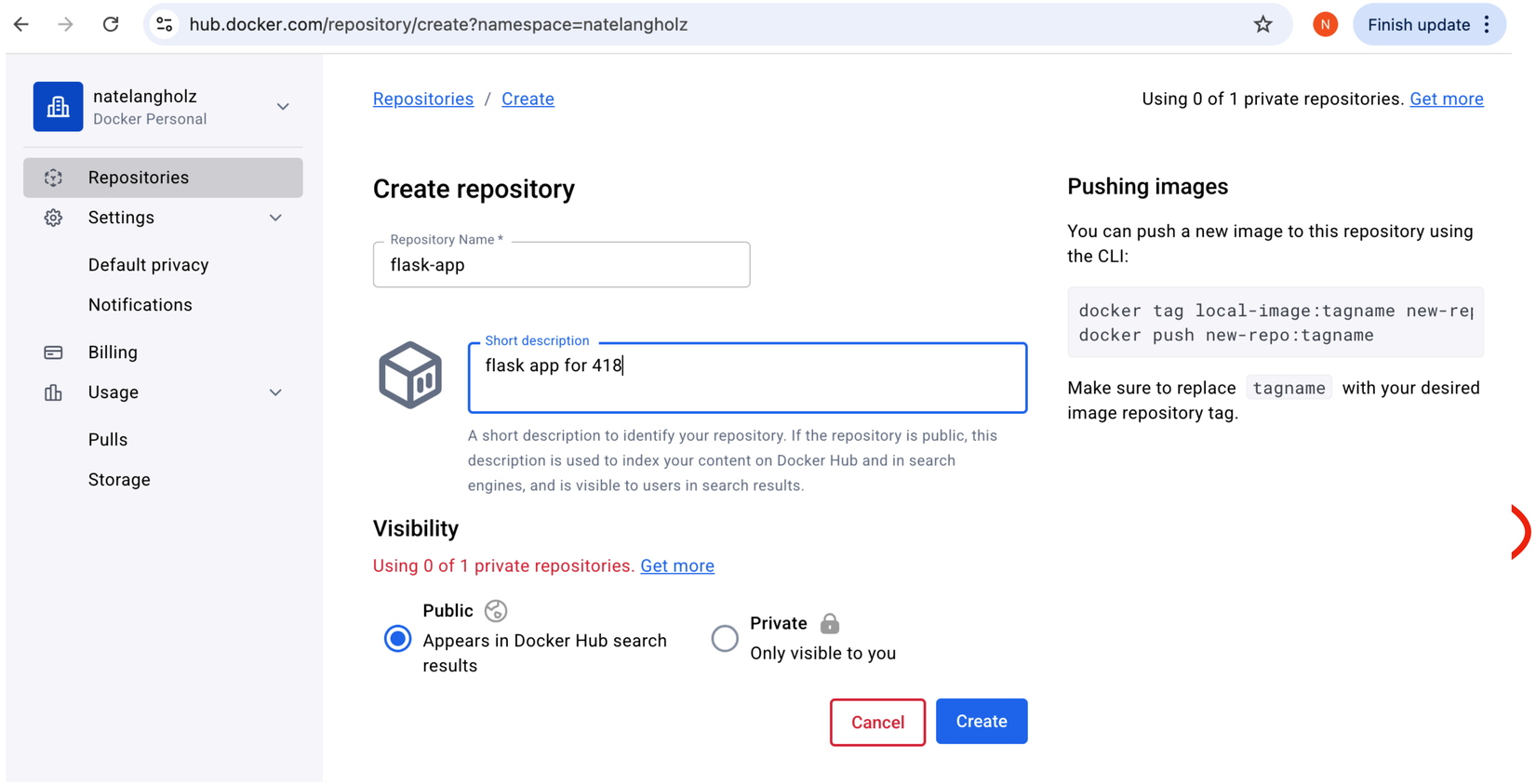
We will need to input a container image url…so we will need to push an image to a container repository (hypothetically could be DockerHub or Artifact Registry)



So far we have kept our docker images and run the containers locally but now we'll need to put them in a image repository

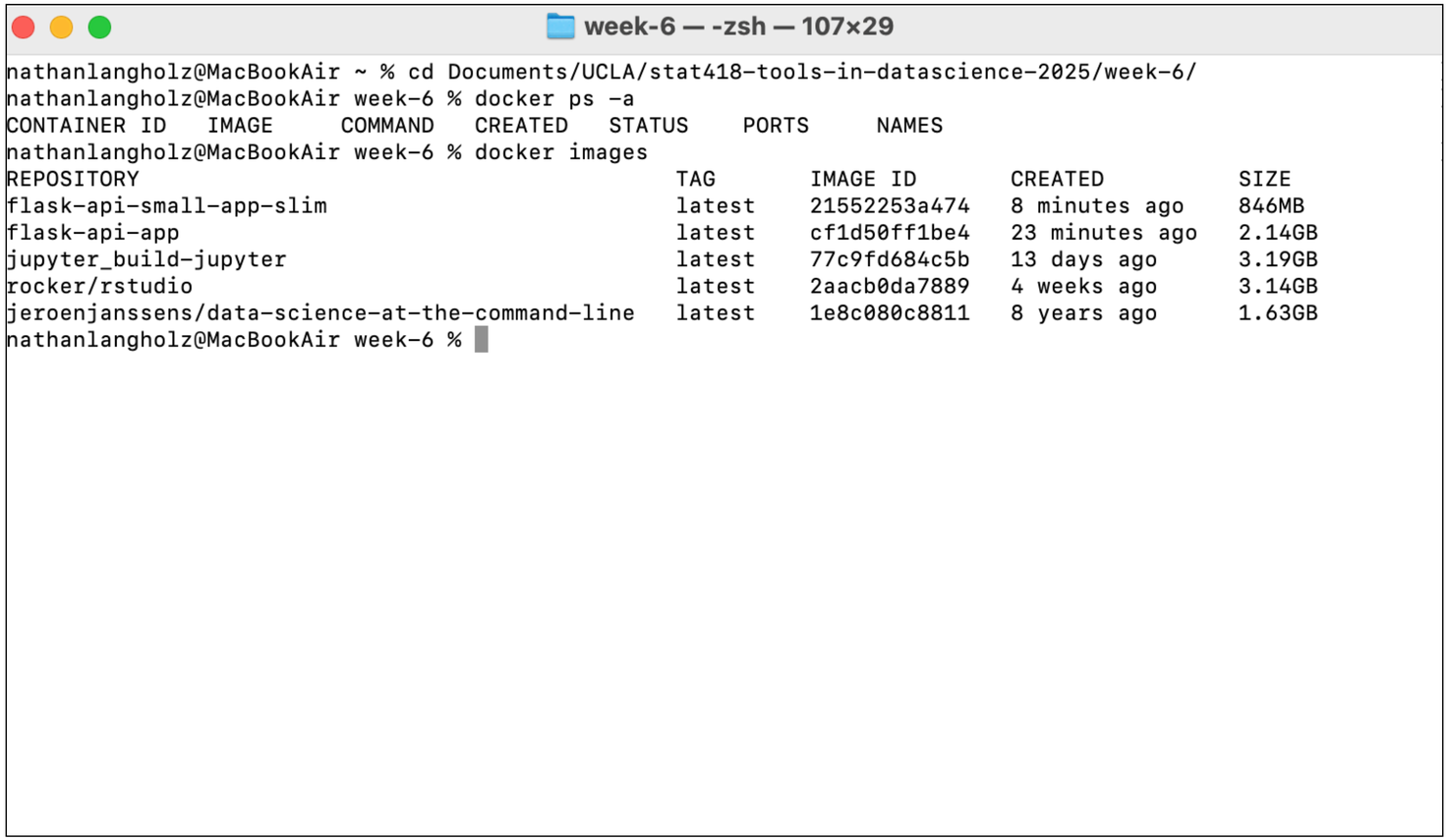


Create a repository where we will push our image..

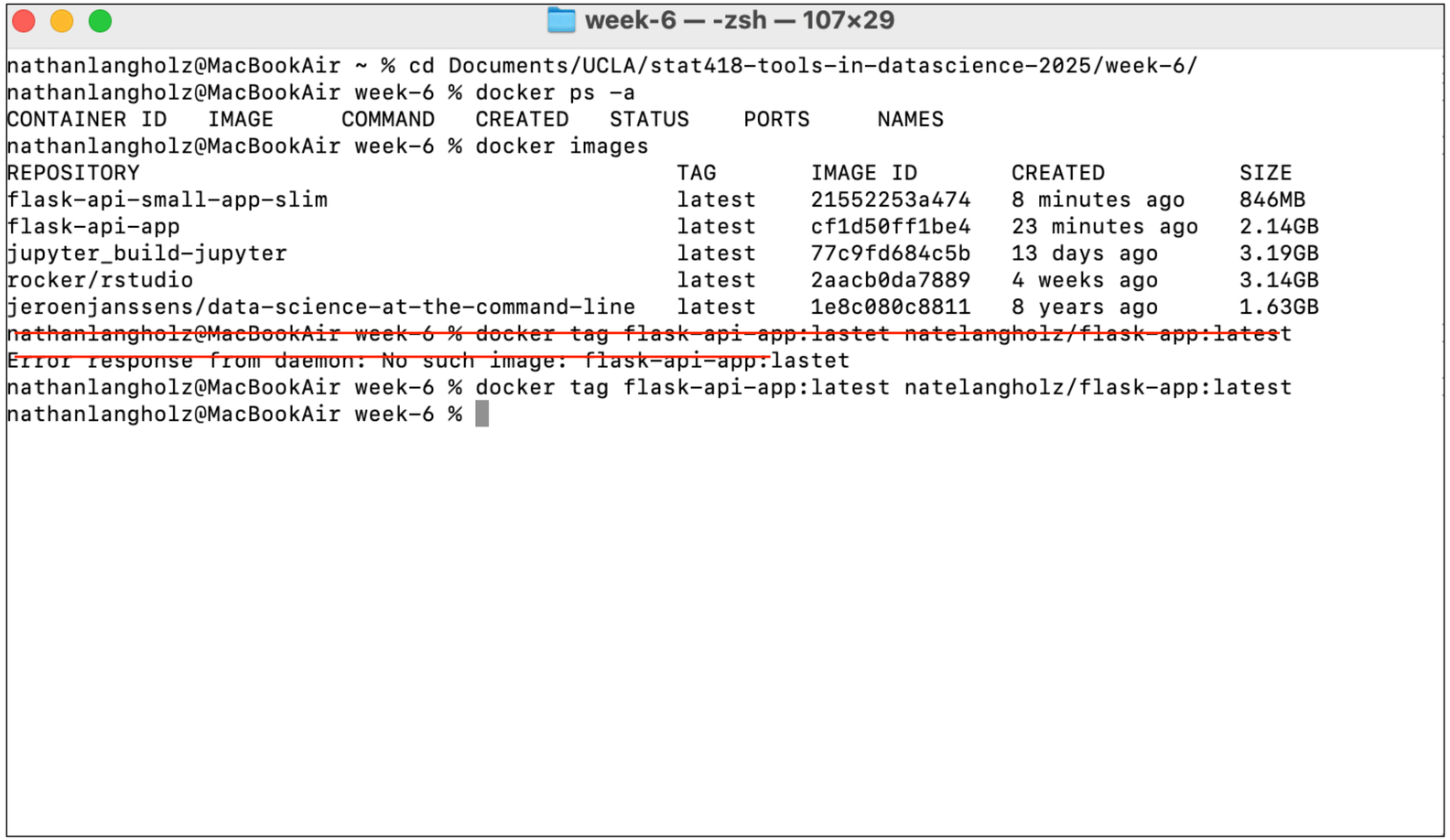


Using the Amazon Linux AMI because its Free tier eligible and the repos include Docker. (I believe the Linux 2 AMI would also work)

What Images do we have available?



Need to change the **tag** of the image to match the repository name. This will take the form of your <username/repository-name:latest>

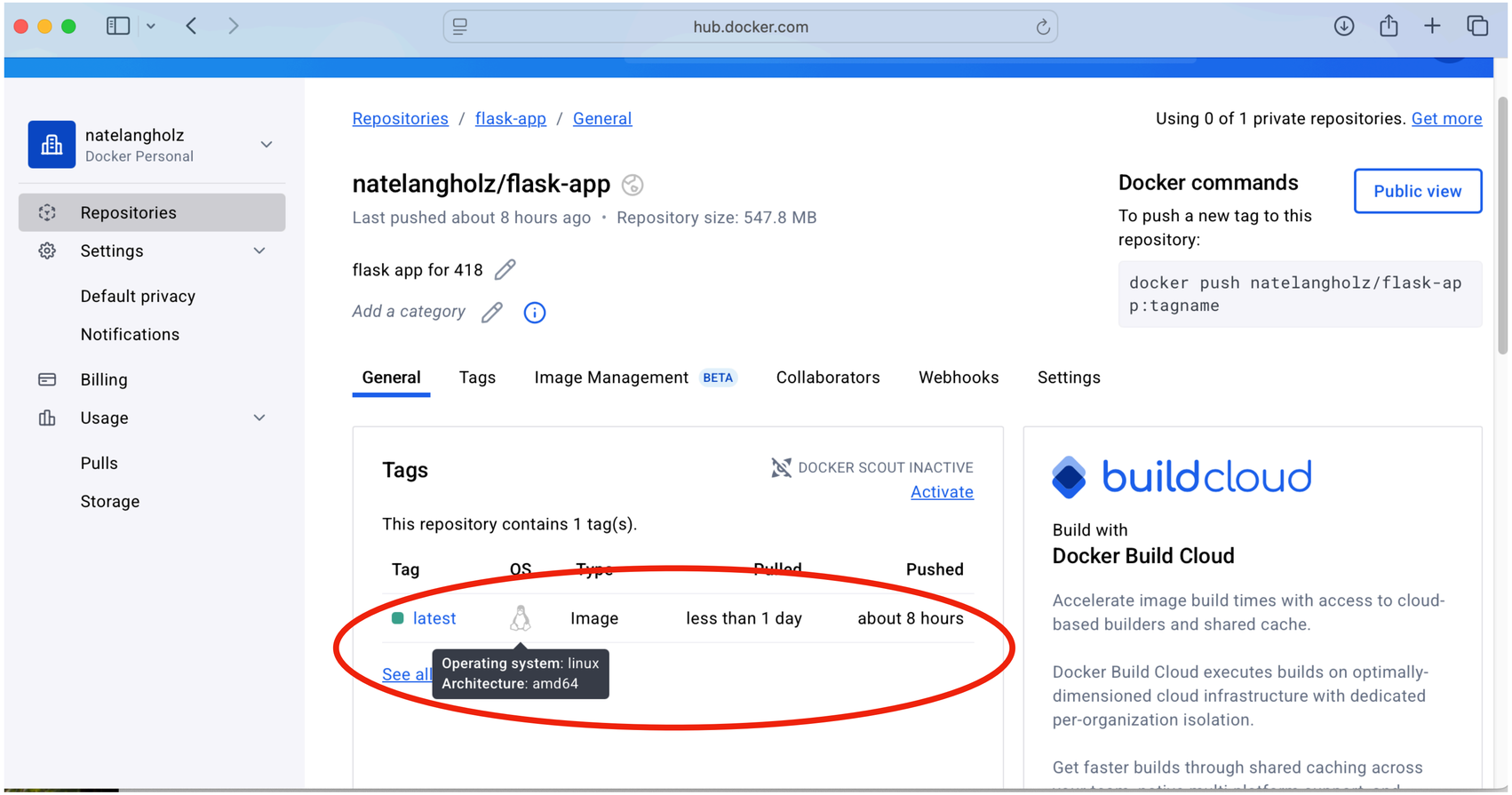


Push the new image to the DockerHub repository

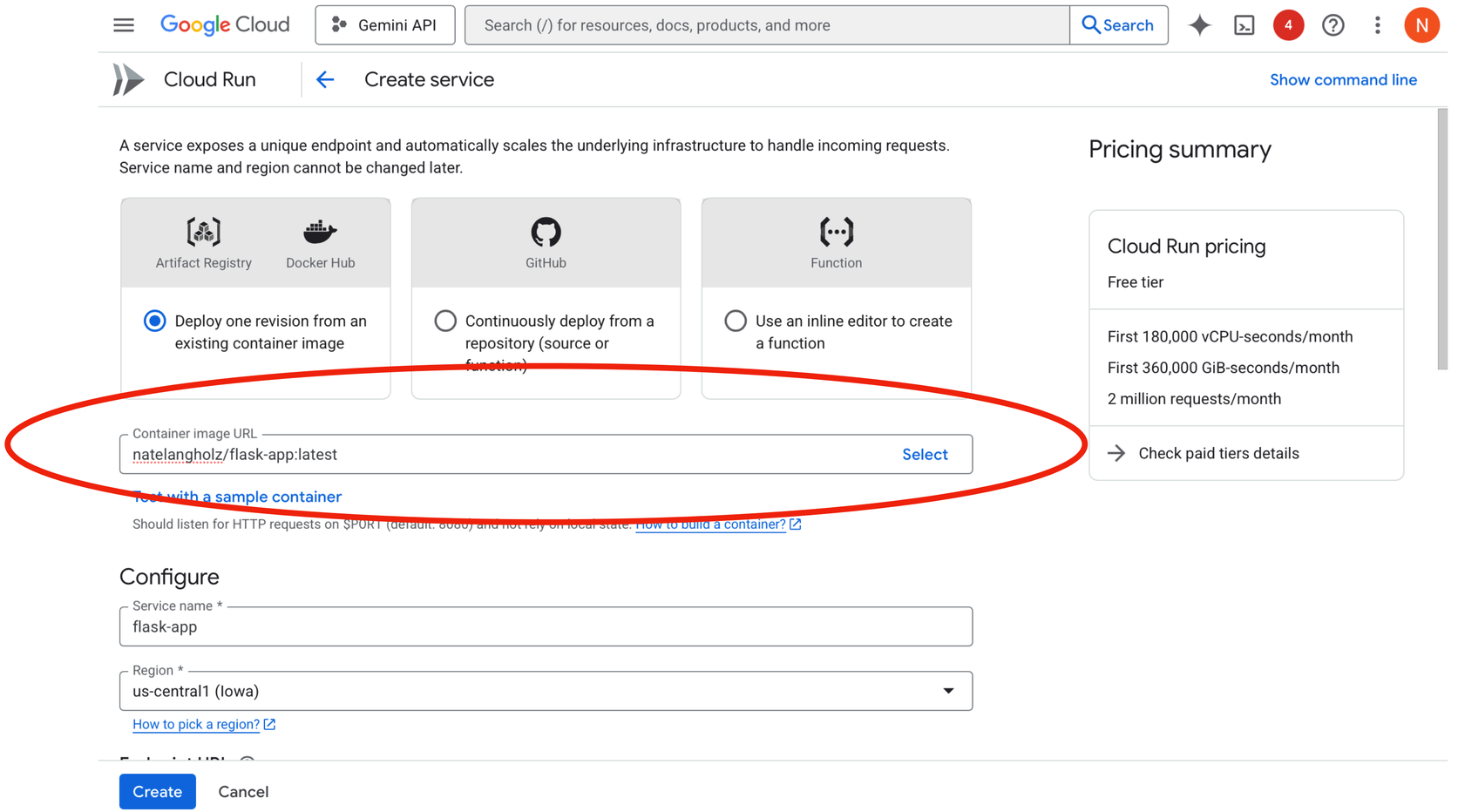
week-6 — docker push natelangholz/flask-app:latest — 107×29

```
nathanlangholz@MacBookAir ~ % cd Documents/UCLA/stat418-tools-in-datascience-2025/week-6/
nathanlangholz@MacBookAir week-6 % docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
nathanlangholz@MacBookAir week-6 % docker images
REPOSITORY                                    TAG       IMAGE ID       CREATED          SIZE
flask-api-small-app-slim                      latest    21552253a474   8 minutes ago    846MB
flask-api-app                                 latest    cf1d50ff1be4   23 minutes ago   2.14GB
jupyter_build-jupyter                         latest    77c9fd684c5b   13 days ago      3.19GB
rocker/rstudio                                latest    2aacb0da7889   4 weeks ago      3.14GB
jeroenjanssens/data-science-at-the-command-line   latest    1e8c080c8811   8 years ago      1.63GB
nathanlangholz@MacBookAir week-6 % docker tag flask-api-app:lastet natelangholz/flask-app:latest
Error response from daemon: No such image: flask-api-app:lastet
nathanlangholz@MacBookAir week-6 % docker tag flask-api-app:latest natelangholz/flask-app:latest
nathanlangholz@MacBookAir week-6 % docker push natelangholz/flask-app:latest
The push refers to repository [docker.io/natelangholz/flask-app]
7a3c21e3aa7f: Pushing  6.162MB/6.162MB
ca513cad200b: Pushing  12.58MB/64.39MB
6cf39de21954: Pushed
6d09090937b0: Pushed
cf05a52c0235: Pushing  10.49MB/48.49MB
8aa2fea2b6be: Pushed
c187b51b626e: Pushing  6.291MB/211.4MB
63964a8518f5: Pushing   7.34MB/24.01MB
1fc0a224894a: Pushed
c7bcf6fd6627: Pushing  11.53MB/25.64MB
52d182f480ab: Pushing  6.291MB/192.8MB
a24b4dd4f057: Pushed
```

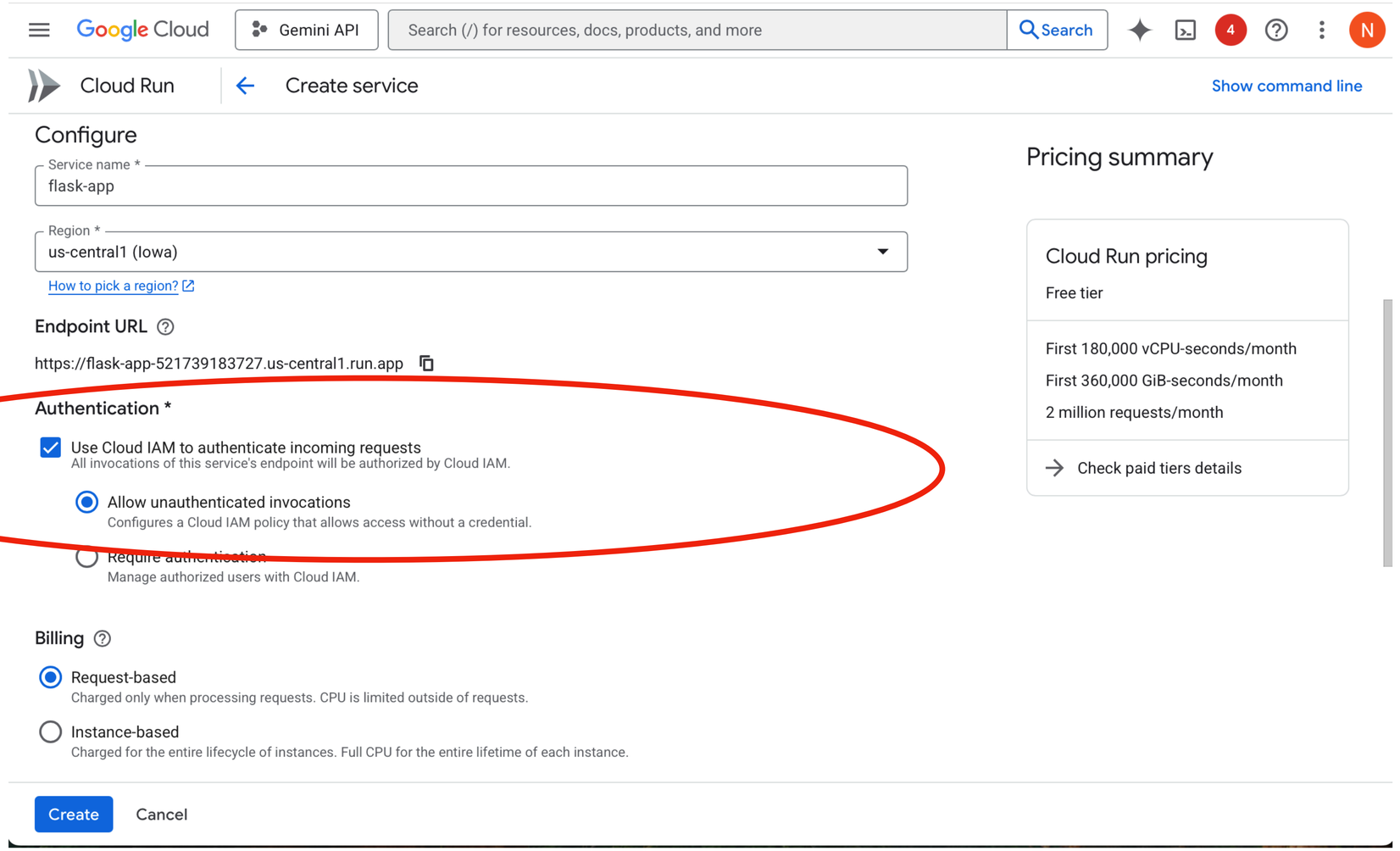
Your DockerHub will now have your pushed image with OS as linux and Type as image



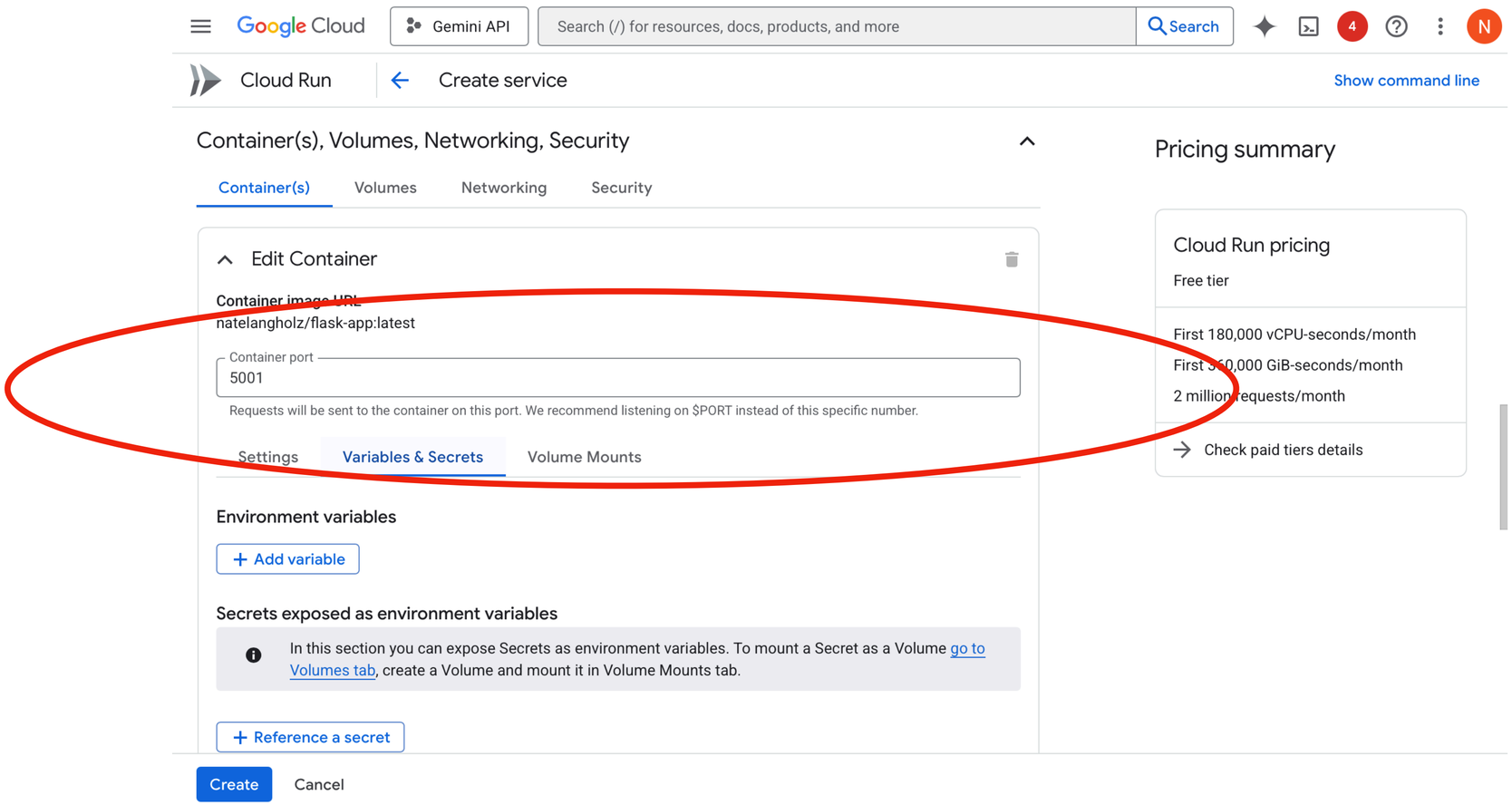
Now back to Google Cloud Run; enter your container image URL



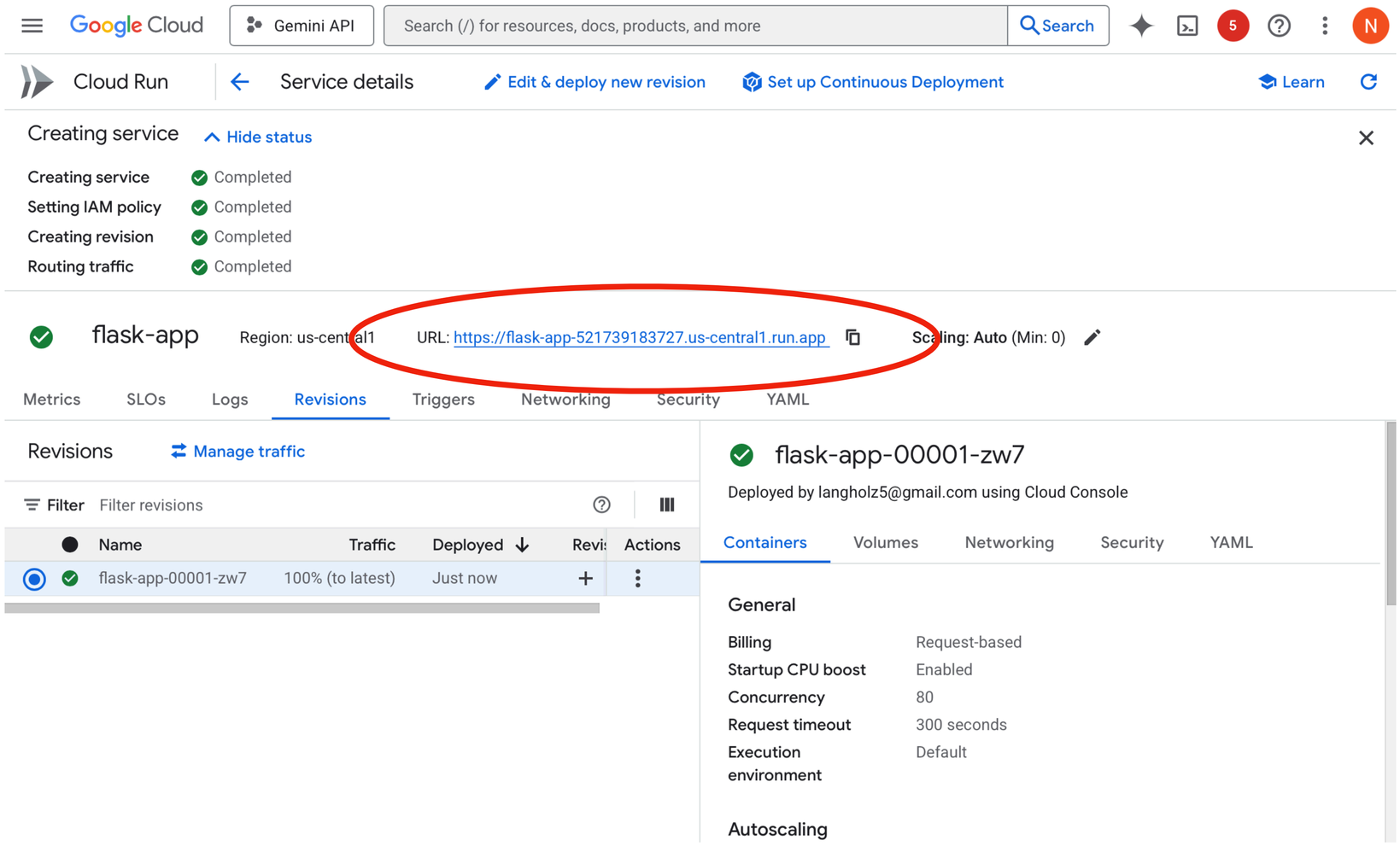
Check **Allow unauthenticated invocations**



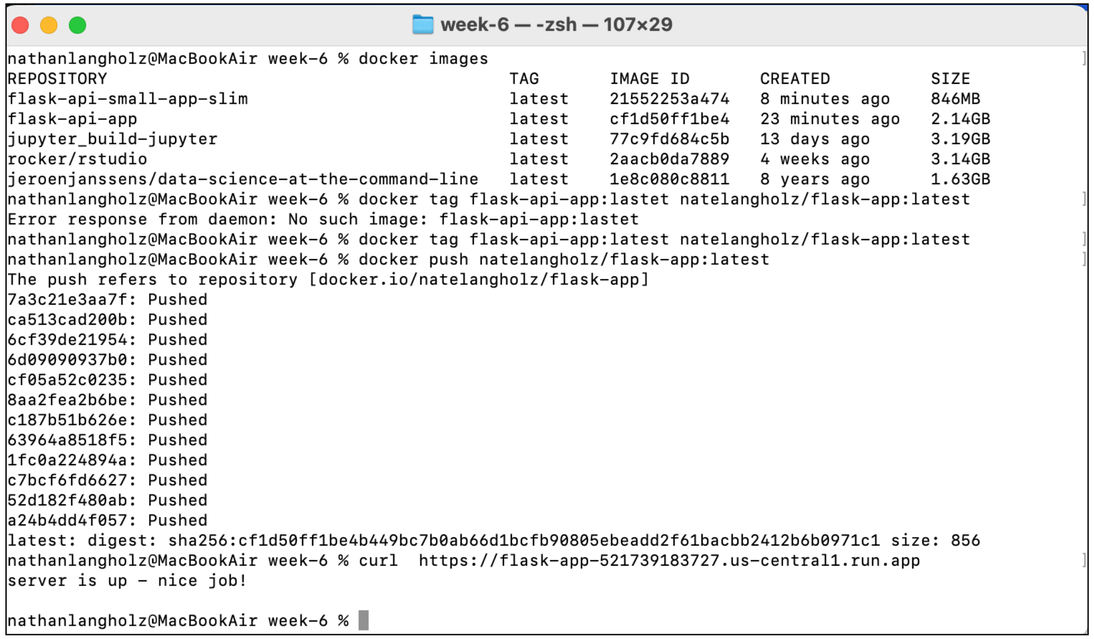
Change the port to match the port you exposed in your Dockerfile, flask app, etc.

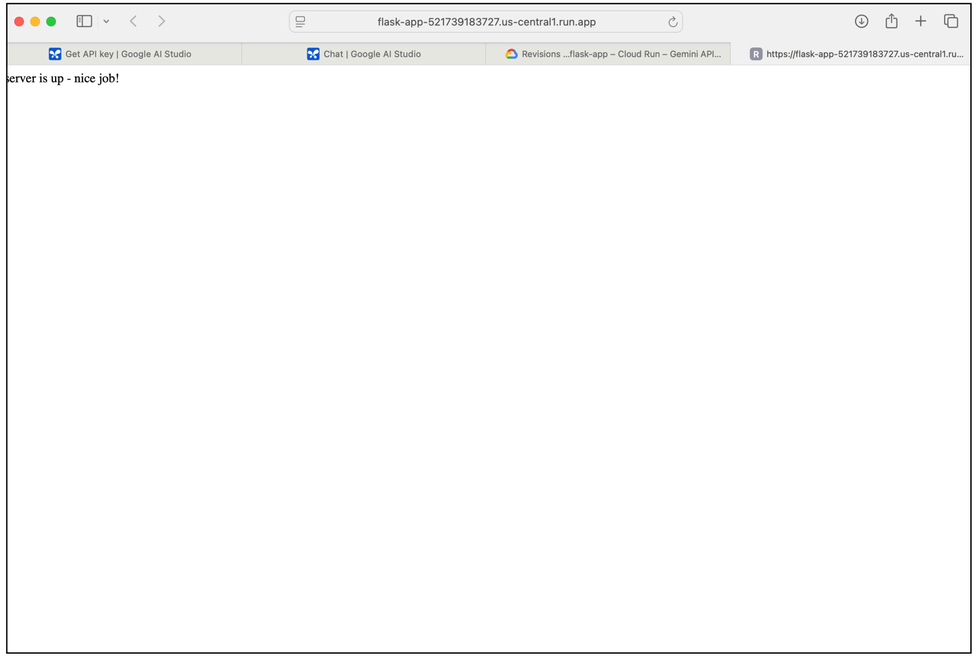


Once you have clicked create and after a few min of deployment you now have a deployed model with the provided endpoint



Can check that the endpoint is up through curl command or simply at the given url.





Now send the JSON test data through the test curl command to the endpoint.
And we should get our prediction back!

week-6 — -zsh — 107×29

```
jeroenjanssens/data-science-at-the-command-line   latest    1e8c080c8811   8 years ago      1.63GB
nathanlangholz@MacBookAir week-6 % docker tag flask-api-app:lastet natelangholz/flask-app:latest
Error response from daemon: No such image: flask-api-app:lastet
nathanlangholz@MacBookAir week-6 % docker tag flask-api-app:latest natelangholz/flask-app:latest
nathanlangholz@MacBookAir week-6 % docker push natelangholz/flask-app:latest
The push refers to repository [docker.io/natelangholz/flask-app]
7a3c21e3aa7f: Pushed
ca513cad200b: Pushed
6cf39de21954: Pushed
6d09090937b0: Pushed
cf05a52c0235: Pushed
8aa2fea2b6be: Pushed
c187b51b626e: Pushed
63964a8518f5: Pushed
1fc0a224894a: Pushed
c7bcf6fd6627: Pushed
52d182f480ab: Pushed
a24b4dd4f057: Pushed
latest: digest: sha256:cf1d50ff1be4b449bc7b0ab66d1bcfb90805ebeadd2f61bacbb2412b6b0971c1 size: 856
nathanlangholz@MacBookAir week-6 % curl  https://flask-app-521739183727.us-central1.run.app
server is up - nice job!

nathanlangholz@MacBookAir week-6 % curl -H "Content-Type: application/json" -X POST -d '{"grade":"10.0","la
t":"47.5396","long":"-122.073","sqft_living":"4495.0","waterfront":"0.0","yr_built":"2007.0"}' "https://fla
sk-app-521739183727.us-central1.run.app/predict_price"
{
  "predict cost": 964695.0817122132
}
nathanlangholz@MacBookAir week-6 %
```

Let’s take a look at making predictions using my api or your neighbors from your machines.