

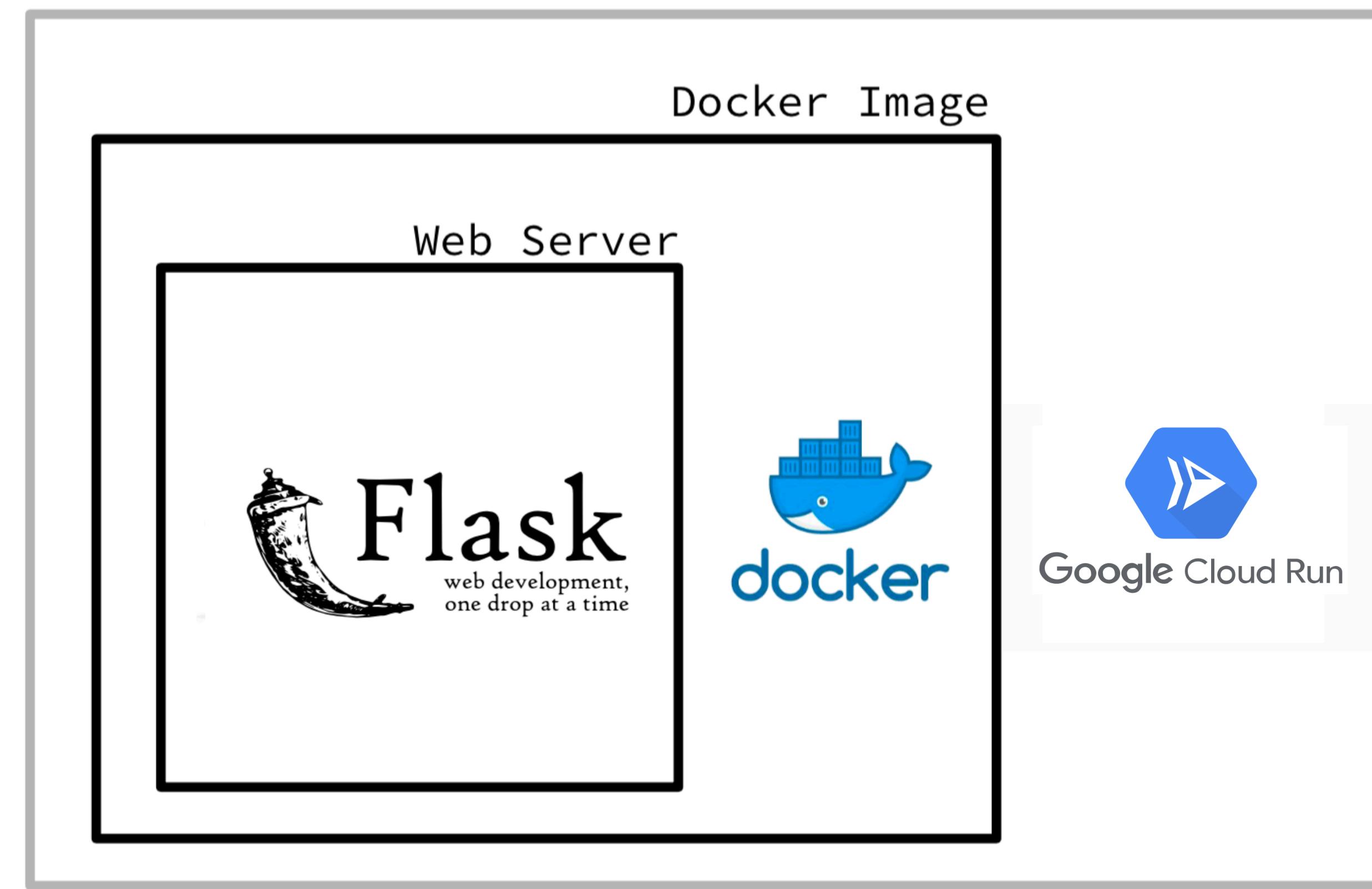
What are we doing here?

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](http://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

[Contact Us](#)[Cloud Run](#)[Features](#)[How It Works](#)[Common Uses](#)[Websites and web applications](#)[AI inference workloads](#)[APIs and microservices](#)[Streaming data processing](#)[Batch data processing](#)[Pricing](#)[Business Case](#)[Partners & Integration](#)

Catch every keynote and exclusive highlight from Google Cloud Next—get your digital pass today.



## 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

## Cloud Run

## Services

+ Deploy container

Connect repo

Write a function



C Release Notes

Services

Jobs

A service exposes a unique endpoint and automatically scales.  
Deploy a container image, source code or a function.

## Service

Each service has a unique endpoint & autoscales deployed code.

## Job

Execute code to completion.

## Services

Filter services

 ● Name Deployment type

Req/sec

Region

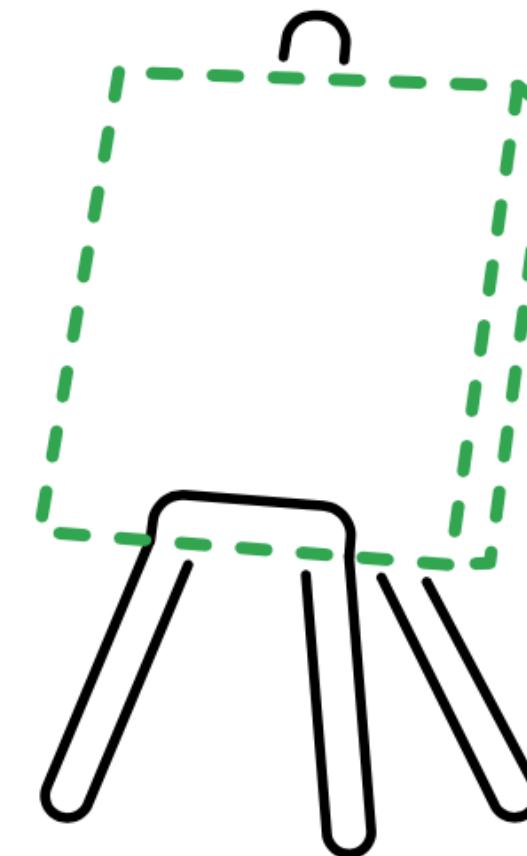
Authentication

Ingress

Recommendation

Last deploy

No results to display



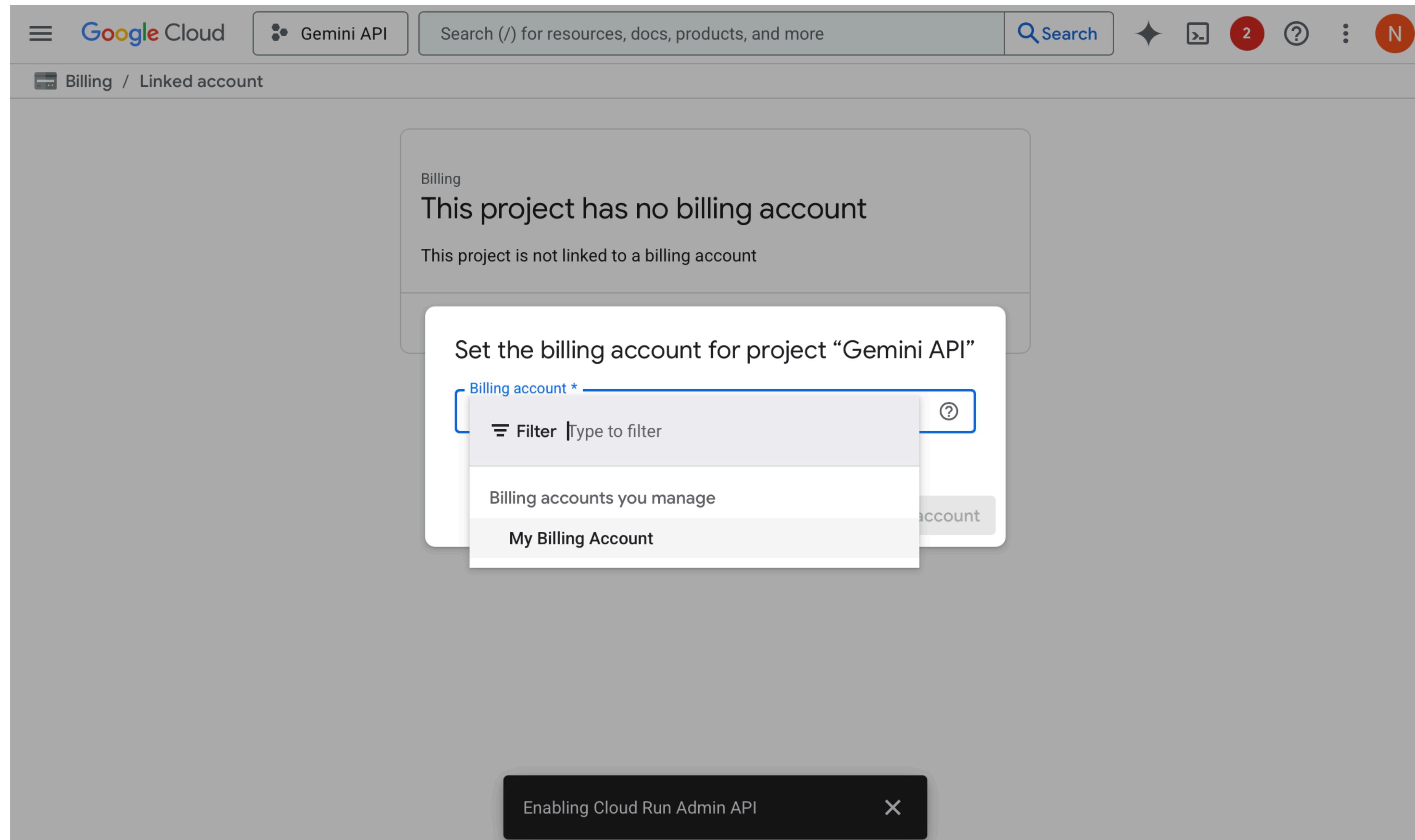
# You will need to enable billing to use

The screenshot shows the Google Cloud Platform interface for creating a new Cloud Run service. The top navigation bar includes the Google Cloud logo, Gemini API, a search bar, and various status indicators. The main header says "Cloud Run" and "Create service". Below this, a sub-header explains that a service exposes a unique endpoint and automatically scales. A note states that the service name and region cannot be changed later.

A central modal dialog box is open, titled "Enable billing to keep using Cloud Run". It contains an informational message: "To use Google Cloud services, you must have a valid Cloud Billing account to verify your identity. No charges are made after this verification process, unless you upgrade to a paid Cloud Billing account." Below this, a link says "To enable billing for this project, go to the billing page. [Learn more](#)". At the bottom of the modal are "Close" and "Go to Billing" buttons.

The main form area has a section titled "Configure" with a "Service name \*" input field. A note below it says "Cloud Run Admin API needs to be enabled to use this option." An "Enable" button is shown next to this note. At the bottom of the form are "Create" and "Cancel" buttons, and a progress bar indicating "Enabling Cloud Run Admin API" with an "X" button to close it.

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)

The screenshot shows the Google Cloud Console interface with the URL `console.cloud.google.com` in the address bar. The page title is "Cloud Run – Gemini API – Google Cloud console". The main navigation bar includes "Google Cloud", "Gemini API", "Search (/) for resources, docs, products, and more", "Search", and "Show command line". Below the navigation, the breadcrumb path is "Cloud Run > Create service".  
  
The main content area is titled "A service exposes a unique endpoint and automatically scales the underlying infrastructure to handle incoming requests. Service name and region cannot be changed later." It features three options:

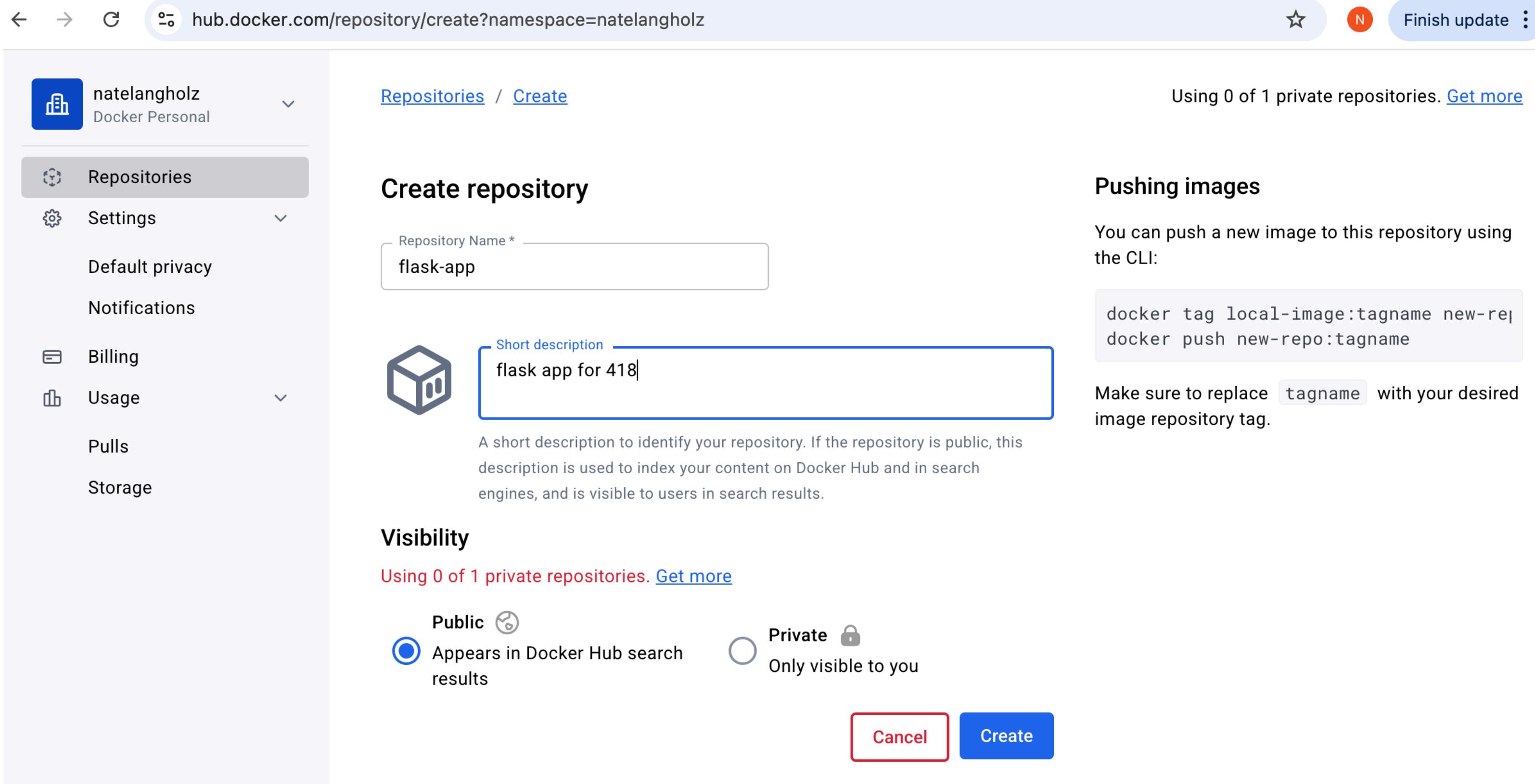
- Artifact Registry** (selected): Deploy one revision from an existing container image.
- Docker Hub**: Continuously deploy from a repository (source or function).
- Function**: Use an inline editor to create a function.

  
A "Container image URL" input field is present, with a "Select" button to its right. Below this is a "Test with a sample container" section, which includes a note: "Should listen for HTTP requests on \$PORT (default: 8080) and not rely on local state." followed by a link "How to build a container?".  
  
The "Configure" section contains fields for "Service name \*" and "Region \*". The "Region" dropdown is set to "us-central1 (Iowa)". A link "How to pick a region?" is provided.  
  
At the bottom, there is an "Endpoint URL" field, a "Create" button, and a "Cancel" button. A confirmation message "Cloud Run Admin API has been enabled" is displayed in a dark overlay window.

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

The screenshot shows the Docker Hub 'My Hub' interface. The top navigation bar includes links for 'hub.docker.com/repositories/natelangholz', a star icon, a notification bell with 'N' notifications, and a 'Finish update' button. Below the header, there's a banner for 'Introducing Docker MCP Catalog and Toolkit - Learn More'. The left sidebar contains links for 'Repositories' (which is selected and highlighted in grey), 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main content area features a large blue 'Welcome to Docker' section with the heading 'Download the desktop application' and options for 'Mac with Intel chip' and 'Mac with Apple chip', both labeled 'MOST COMMON'. Below this, it says 'Also available for [Windows](#) and [Linux](#)'. At the bottom, there are three cards: 'Create a Repository' (with a red oval drawn around it), 'Docker Hub Basics', and 'Language-Specific Guides'. The URL 'https://hub.docker.com' is visible at the bottom left of the page.

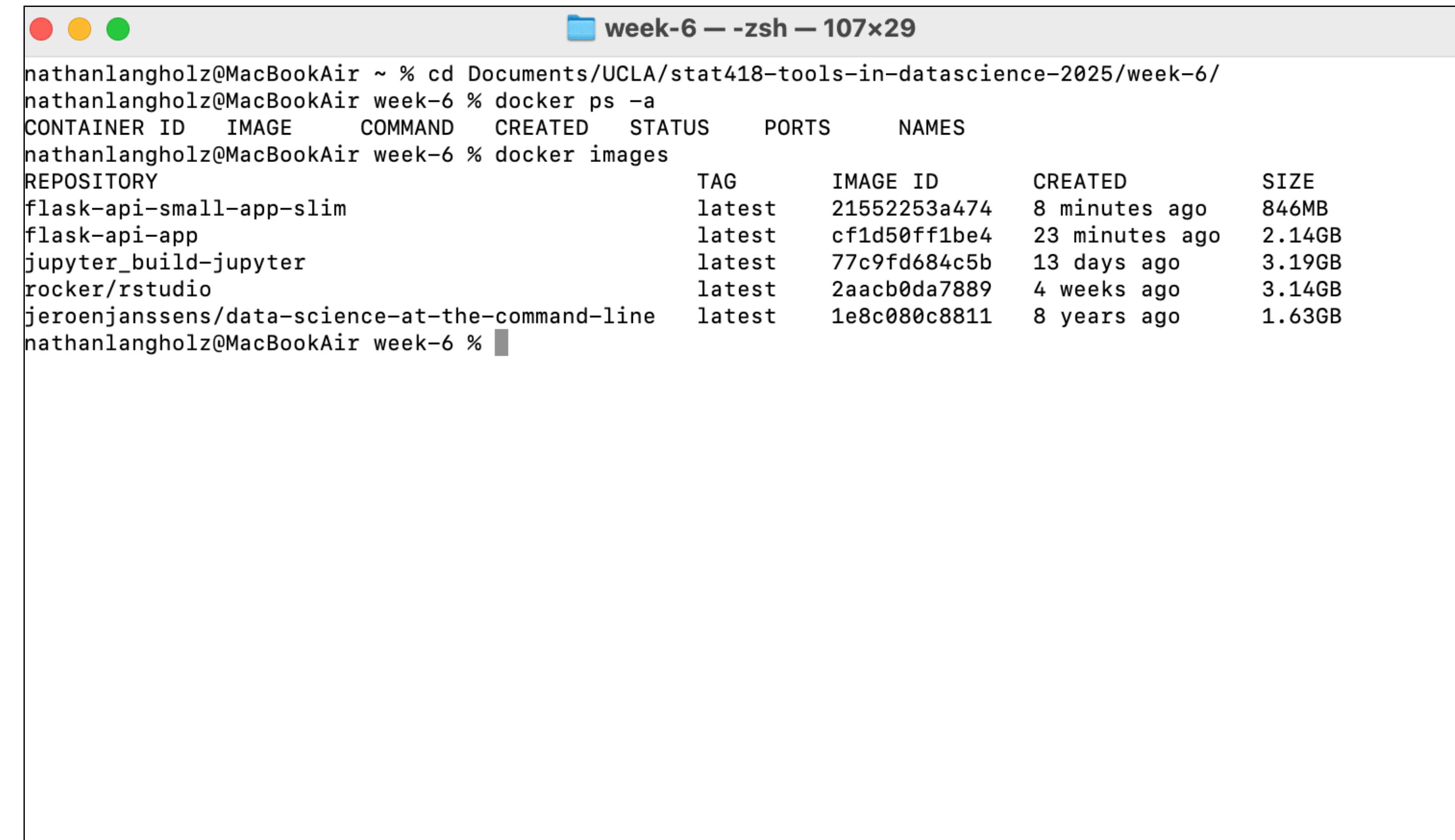
# Create a repository where we will push our image..



The screenshot shows the Docker Hub repository creation interface. The URL in the address bar is `hub.docker.com/repository/create?namespace=natelangholz`. The left sidebar shows the user's profile: **natelangholz** (Docker Personal) with options for Repositories, Settings, Default privacy, Notifications, Billing, Usage, Pulls, and Storage. The main area is titled "Create repository". The "Repository Name" field contains "flask-app". The "Short description" field contains "flask app for 418". Below these fields, a note says: "A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results." Under "Visibility", there are two options: "Public" (selected, indicated by a blue circle) and "Private" (indicated by a grey circle). The "Public" option includes the sub-note "Appears in Docker Hub search results". At the bottom right are "Cancel" and "Create" buttons. The top right of the page shows "Using 0 of 1 private repositories. [Get more](#)". On the far right, there is a red curved arrow pointing right.

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?



The screenshot shows a terminal window titled "week-6 -- zsh -- 107x29". The terminal output displays the results of running Docker commands to list available images.

```
nathanlangholz@MacBookAir ~ % cd Documents/UCLA/stat418-tools-in-datasience-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 %
```

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

```
week-6 — zsh — 107x29

nathanlangholz@MacBookAir ~ % cd Documents/UCLA/stat418-tools-in-datasience-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:latest natelangholz/flask-app:latest
Error response from daemon: No such image: flask-api-app.latest
nathanlangholz@MacBookAir week-6 % docker tag flask-api-app:latest natelangholz/flask-app:latest
nathanlangholz@MacBookAir week-6 %
```

## Push the new image to the DockerHub repository

```
week-6 — docker push natelangholz/flask-app:latest — 107x29
nathanlangholz@MacBookAir ~ % cd Documents/UCLA/stat418-tools-in-datasience-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:latest natelangholz/flask-app:latest
Error response from daemon: No such image: flask-api-app:latest
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

The screenshot shows a DockerHub repository page for the user 'natelangholz'. The repository is named 'flask-app' and has a single tag, 'latest'. A red oval highlights the table row for the 'latest' tag, which includes columns for Tag, OS, Type, Pulled, and Pushed. The 'OS' column shows 'linux', and the 'Type' column shows 'Image'. Below the table, a tooltip provides detailed information: 'Operating system: linux' and 'Architecture: amd64'.

hub.docker.com

Repositories / flask-app / General

Using 0 of 1 private repositories. [Get more](#)

**Docker commands**

To push a new tag to this repository:

```
docker push natelangholz/flask-app :tagname
```

**General** Tags Image Management BETA Collaborators Webhooks Settings

**Tags**

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest	linux	Image	less than 1 day	about 8 hours

See all

Operating system: linux  
Architecture: amd64

**buildcloud**

Build with  
**Docker Build Cloud**

Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

Get faster builds through shared caching across

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

The screenshot shows the Google Cloud Run 'Create service' interface. At the top, there's a navigation bar with 'Google Cloud' and 'Gemini API' tabs, a search bar, and various icons. Below the navigation, the title 'Cloud Run' and 'Create service' are displayed, along with a 'Show command line' link.

A service exposes a unique endpoint and automatically scales the underlying infrastructure to handle incoming requests. Service name and region cannot be changed later.

**Pricing summary**

**Cloud Run pricing**

Free tier

- First 180,000 vCPU-seconds/month
- First 360,000 GiB-seconds/month
- 2 million requests/month

→ Check paid tiers details

**Deploy options:**

- Artifact Registry (selected)
- Docker Hub
- GitHub
- Function

**Container image URL:** natelangholz/flask-app:latest

**Select**

**Configure**

Service name \* flask-app

Region \* us-central1 (Iowa)

How to pick a region? ↗

Create Cancel

# Check Allow unauthenticated invocations

Google Cloud Gemini API Search (/) for resources, docs, products, and more Search 4 N

Cloud Run Create service Show command line

**Configure**

Service name \* flask-app

Region \* us-central1 (Iowa)

[How to pick a region?](#)

**Endpoint URL** <https://flask-app-521739183727.us-central1.run.app>

**Authentication \***

Use Cloud IAM to authenticate incoming requests  
All invocations of this service's endpoint will be authorized by Cloud IAM.

Allow unauthenticated invocations  
Configures a Cloud IAM policy that allows access without a credential.

Require authentication  
Manage authorized users with Cloud IAM.

**Billing**

Request-based  
Charged only when processing requests. CPU is limited outside of requests.

Instance-based  
Charged for the entire lifecycle of instances. Full CPU for the entire lifetime of each instance.

Create Cancel

**Pricing summary**

Cloud Run pricing

Free tier

First 180,000 vCPU-seconds/month

First 360,000 GiB-seconds/month

2 million requests/month

[Check paid tiers details](#)

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

The screenshot shows the Google Cloud Cloud Run 'Create service' interface. At the top, there's a navigation bar with the Google Cloud logo, a Gemini API button, a search bar, and various status indicators. Below the navigation, the page title is 'Cloud Run' and the sub-action is 'Create service'. On the left, there's a sidebar titled 'Container(s), Volumes, Networking, Security' with tabs for 'Container(s)', 'Volumes', 'Networking', and 'Security'. The 'Container(s)' tab is active, showing an 'Edit Container' section. Inside this section, the 'Container image URL' is set to 'natelangholz/flask-app:latest' and the 'Container port' is set to '5001'. A note below says, 'Requests will be sent to the container on this port. We recommend listening on \$PORT instead of this specific number.' Below this, there are tabs for 'Settings', 'Variables & Secrets' (which is selected), and 'Volume Mounts'. The 'Variables & Secrets' tab has a '+ Add variable' button. Underneath, there's a section for 'Secrets exposed as environment variables' with a note about mounting secrets as volumes. At the bottom of the main content area are 'Create' and 'Cancel' buttons. To the right of the main content, there's a 'Pricing summary' box titled 'Cloud Run pricing' under the 'Free tier'. It lists: 'First 180,000 vCPU-seconds/month', 'First 50,000 GiB-seconds/month', and '2 million requests/month'. A link 'Check paid tiers details' is also present. A large red oval highlights the 'Container port' input field.

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

The screenshot shows the Google Cloud Run Service details page for a service named "flask-app". The top navigation bar includes the Google Cloud logo, Gemini API, a search bar, and various icons. Below the header, there's a progress section titled "Creating service" with four completed steps: "Creating service" (Completed), "Setting IAM policy" (Completed), "Creating revision" (Completed), and "Routing traffic" (Completed). The main service card for "flask-app" shows it is in "us-central1" region, with a URL link (<https://flask-app-521739183727.us-central1.run.app>) and scaling set to "Auto (Min: 0)". A red oval highlights the URL link. The "Revisions" tab is selected, showing one revision named "flask-app-00001-zw7" which was deployed "Just now". The revision card indicates it was deployed by "langholz5@gmail.com using Cloud Console". The right panel displays the "Containers" configuration for the revision, including settings for General, Billing, Startup CPU boost, Concurrency, Request timeout, Execution environment, and Autoscaling.

Creating service [Hide status](#) X

Creating service ✓ Completed  
Setting IAM policy ✓ Completed  
Creating revision ✓ Completed  
Routing traffic ✓ Completed

✓ flask-app Region: us-central1 URL: <https://flask-app-521739183727.us-central1.run.app> Scaling: Auto (Min: 0) Edit

Metrics SLOs Logs Revisions Triggers Networking Security YAML

Revisions Manage traffic

✓ flask-app-00001-zw7 Deployed by langholz5@gmail.com using Cloud Console

Name	Traffic	Deployed	Revision	Actions
flask-app-00001-zw7	100% (to latest)	Just now	<span style="color: green;">✓</span>	<span style="color: green;">+</span> <span style="color: green;">⋮</span>

Containers Volumes Networking Security YAML

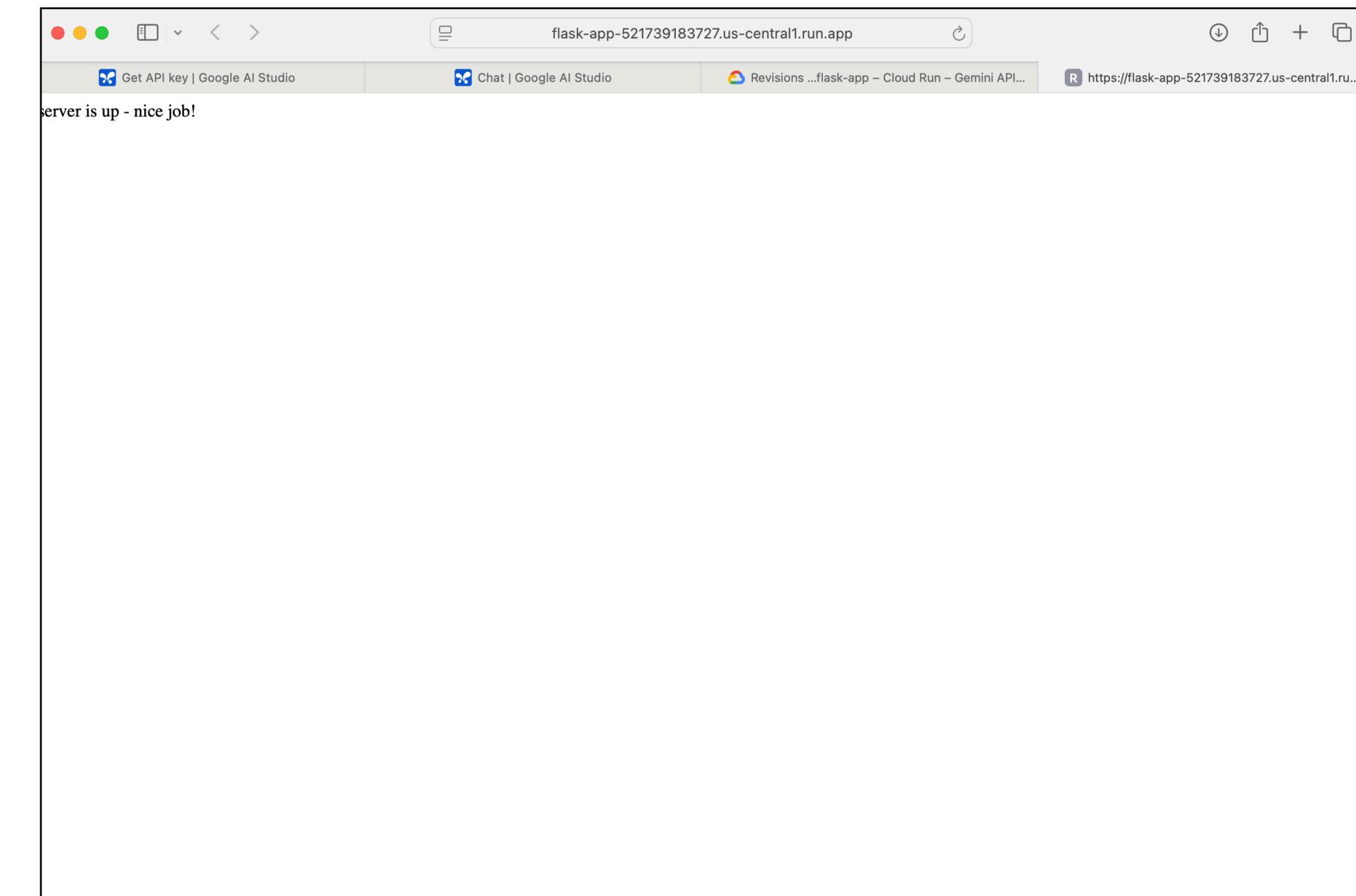
**General**

Billing Request-based  
Startup CPU boost Enabled  
Concurrency 80  
Request timeout 300 seconds  
Execution environment Default

Autoscaling

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

```
week-6 -- zsh -- 107x29
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: 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 %
```



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

```
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","lat":47.5396,"long":-122.073,"sqft_living":4495.0,"waterfront":0.0,"yr_built":2007.0}' "https://flask-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.