Cloudflare Hiring Assignment

**November 02, 2021**

**Overview**

*‘In this exercise, you'll learn to use Cloudflare Workers and Pages by creating your very own social media platform. The instructions below will guide you through the requirements.’ (https://apply.cloudflareworkers.com/general)*

1. **Focus:** There are 2 parts:
   1. Frontend Web App
      1. Deploy a web app using **Cloudflare Pages**
         1. Using a template such as **Create-React-App**
      2. Inside the web app
         1. Get the posts from your API & organize however I choose
         2. Try to emulate the feel of a real social media platform
      3. Make some posts!
         1. Test the app by creating a few posts w\ some information about yourself
   2. Backend API
      1. Using Cloudflare Workers to launch a serverless API
         1. Create a new Workers project w\ **Wrangler**
            1. Allows creation & storage of new posts
            2. Distributes these posts to visitors of site
      2. Storing a collection of posts
         1. Using **Workers KV** namespace
            1. Simple key-value cloud storage environment
2. **Outline of Main Elements:**
   1. API Endpoints
      1. GET /posts
         1. Receives JSON response containing a list of post objects
            1. From KV namespace
         2. Sample output:
            1. [{“title”: ”My first post”, “username”: ”coolguy123”, “content”: “Hey!”},{“title”: ”Storytime”, “username”: ”kn0thing”, “content”: “So, the other day…”}]
      2. POST /posts
         1. Serves as the way to create a new post: taking JSON input.
            1. {“title”: ”My first post”, “username”: ”coolguy123”, “content”: “Hey!”}
         2. Meaningful error message & status code should be returned if there are any problems;
            1. Otherwise, return ‘**success**’.
   2. Extra Credit
      1. Showcase talent by including extra features:
         1. User Posts
            1. Adding a series of inputs to your frontend which will fill out the necessary info for a post & submit that info to your POST endpoint to be added to the website.
         2. Content Variety
            1. Include extra content – photos, links, GIFs & other multimedia forms
            2. Add configuration options to each post allowing it to be presented differently depending on the content.

Give examples for each type of post.

* + - 1. Interactivity
         1. Adding different types of content for user interaction

Voting systems ( upvote \ downvote )

Emoji

Comment sections

# **Submitting your assignment** Publish it!

When done, publish it on **Workers** and **Pages**! After following ‘Get Started’, you’ll receive an email with submission instructions:

* Clone the git repository generated for your general assignment
* Copy the files from your Worker’s project folder into ‘workers’
* Copy the files from our Pages project folder into ‘pages’
* In the “info.txt” file, paste the URL to your published workers.dev project directly after the prompt (on the same line, leaving a space)
  + Repeat for your published pages.dev URL
* Add everything, commit & push!

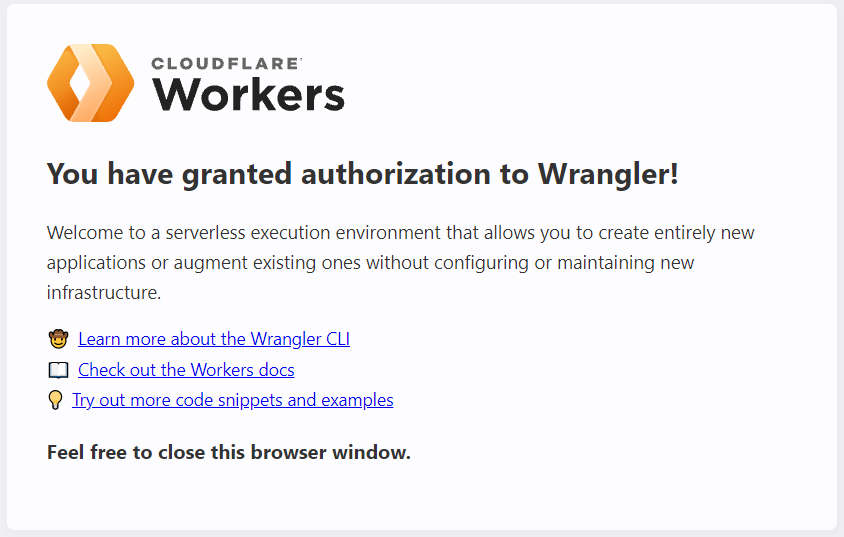
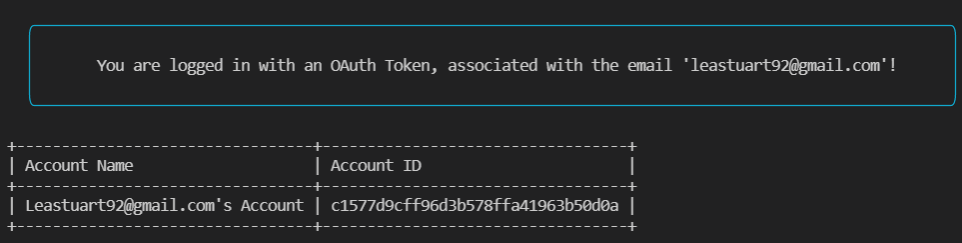
# **Get Started** Introduction

To start I decided to begin with the Backend portion. First I set up the environment and then generated a new Worker project.

Getting started a few things need to be installed to get the environment setup:

* npm
  + Installed version: 8.0.0
* npm install -g @cloudflare/wrangler
  + wrangler –version
    - Installed version: 1.19.4

Then, you need to authenticate wrangler installation with cloudflare user:

* wrangler login
  + Allow Wrangler to open a page in browser [ Y ]
  + Within the browser:
    - Click ‘Allow’ to authenticate
    - 
  + wrangler whoami
    - 

# **Generating Worker** Starting a new Worker project

# Next I generated a new Worker project:

* wrangler generate my-worker
  + ‘my-worker’ == name of the project
* Extra templates and documentation can be found at:
  + developers.cloudflare.com/workers/starters
  + developers.cloudflare.com/workers/runtime-apis/response
* After generating can preview project:
  + wrangler preview
    - A successful preview will say: ‘Your Worker responded with: Hello worker!’

From here you can already deploy & publish your workers project:

* Run ‘wrangler whoami’
  + Should get a print out like before
  + Copy the ‘Account ID’
* Open ‘wrangler.toml’
  + Paste the ‘Account ID’
  + \*\* SPECIAL NOTE \*\*
    - It is alright to have the account ID in plain text because without the API token you cannot do anything with it.
    - ‘Route’ & ‘ZoneID’ allow you to deploy to a specific domain rather than the default Cloudflare ‘workers\_dev’ domain.
* Run ‘wrangler dev’
  + Launches the developing environment
  + Ctrl + C closes the wrangler dev
* Run ‘wrangler publish’
  + Publishes the project

# **Render HTML with Cloudflare Workers** Render website templates

# Next I created a new file:

* template.js – to set up a HTML template
  + Using a worker: return a string
    - const template = () => ` < HTML GOES HERE > `
    - End of template.js make sure you ‘export default template’
* index.js – renders HTML
  + Import template: import template from ‘./template’
  + IN THE RESPONSE function:
    - return new Response(template(), {headers: {‘content-type’: ‘text/html’ }
      * Make sure to change the content-type or it will print out plain text to screen;
* wrangler.toml
  + Change the type from ‘javascript’ to ‘webpack’
    - Allows you to import outside modules
* Finally ‘wrangler publish’ your project

# **Render Cloudflare Region Data** For a Request using request.cf

# Using the request argument:

* request.cf – specific to Cloudflare.
  + Has a lot of features:
    - Shows *where* the user is requesting from
    - What data center their request is being served from
    - Documentation: developers.cloudflare.com/workers/runtime-apis/requests#incomingrequestcfproperties
    - Windows doesn’t support the emoji

# **Deploy Custom Domain** Using Cloudflare Wrangler

# If you have a specific domain you wish to use for your application, you can deploy it using Wrangler (i.e.custom domain for personal website). ***IMPORTANT*** *– you must have your domain inside Cloudflare.*

After you choose your domain:

* Choose it from your list of available domains on Cloudflare
* Once inside the dashboard scroll down to **API**
  + 2 things are needed:
    - Zone ID
    - Route

# Inside the *wrangler.toml*:

* Set up production environment by putting : [ env.production ] above route & zone\_id;
  + Set workers\_dev = false;
  + To publish to production → **wrangler publish -e production**
* zone\_id = “ “
* route = “https://customedomain.com/\* ”
  + A wildcard ( \* ) means anything inside https://customedomain.com/ will use this worker;

Finally, go back into the dashboard and set up a DNS record:

* DNS records are sets of instructions that live on DNS servers - vital to the success of a DNS lookup
  + https://www.cloudflare.com/learning/dns/dns-records/
* ‘Add Record’ button & follow prompts.

# **Create a React App** Frontend Application

# In the outer folder create a new folder to house the React Frontend. Then inside that folder use the terminal:

* npx create-react-app my-app
  + ‘My-app’ is the name of your application
* cd my-app
* npm start

Delete files you don’t need and start from a blank ‘App.js’. Create a function to make a request to the Workers API you created before.

const getImages = async query => {

const url = “<https://my-worker.lrieger.workers.dev/>”;

const resp = await fetch(url, {

method: “POST”;

body: JSON.stringify({ query }),

headers: {‘Content-Type’: ‘application/json’},

})

return resp.json();

}