

Application Services Assignment

Purpose: This assignment is intended to evaluate your hands-on capabilities, give you a taste for the work we do, and show us how you communicate complex subjects in writing. You can refer to: Application Services, Cloudflare Docs and the Cloudflare Dashboard.

1. Project Overview

Project Name: Application Services Assignment

Created By: Micael Santos

Date: 18/02/2026

Cloud Provider: CloudFlare/AWS

2. Introduction

This document describes the implementation of a secure web application architecture using Cloudflare Application Services, as requested in the assignment.

The solution integrates multiple Cloudflare products including DNS, SSL/TLS, Cloudflare Tunnel, Zero Trust Access, Workers and R2 Storage, together with an AWS EC2 origin environment running a Flask application behind an NGINX reverse proxy.

The goal was to demonstrate the ability to design, deploy and secure an application using Cloudflare services while following best practices and documenting the process clearly.

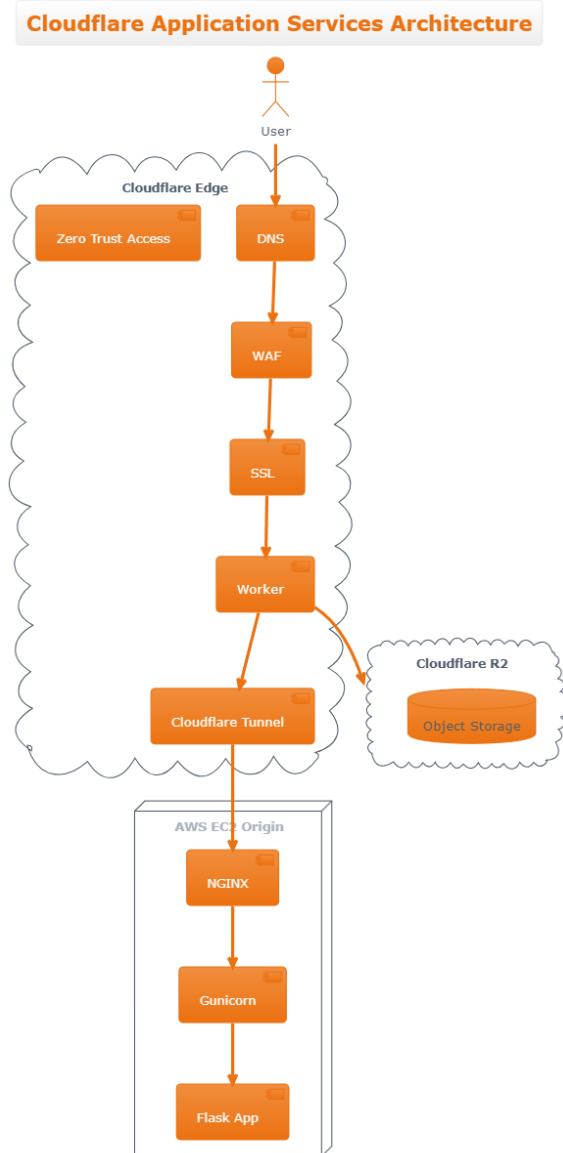


Fig. 1 Architecture diagram

3. Architecture Overview

The implemented architecture consists of two main traffic flows:

1. Public Application Flow:

User → Cloudflare Edge → Cloudflare Tunnel → AWS EC2 → NGINX → Gunicorn → Flask Application

Public Flow - Headers Application (Cloudflare > Tunnel -> EC2)

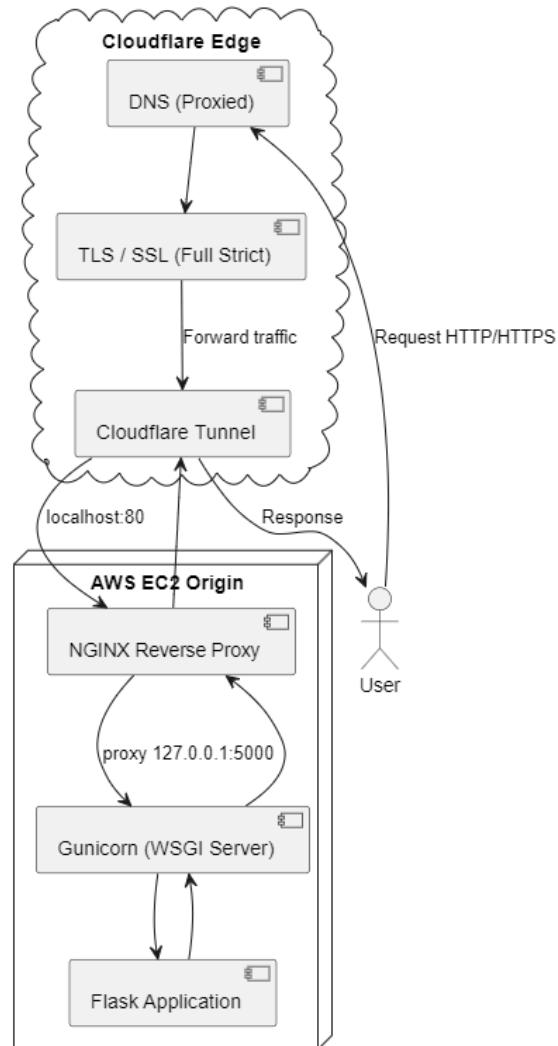


Fig. 2Public diagram

2. Secure Application Flow:

User → Cloudflare Edge → Zero Trust Access → Cloudflare Worker → Private R2 Bucket

The origin server is not directly exposed to the Internet, as all traffic is routed through Cloudflare Tunnel.

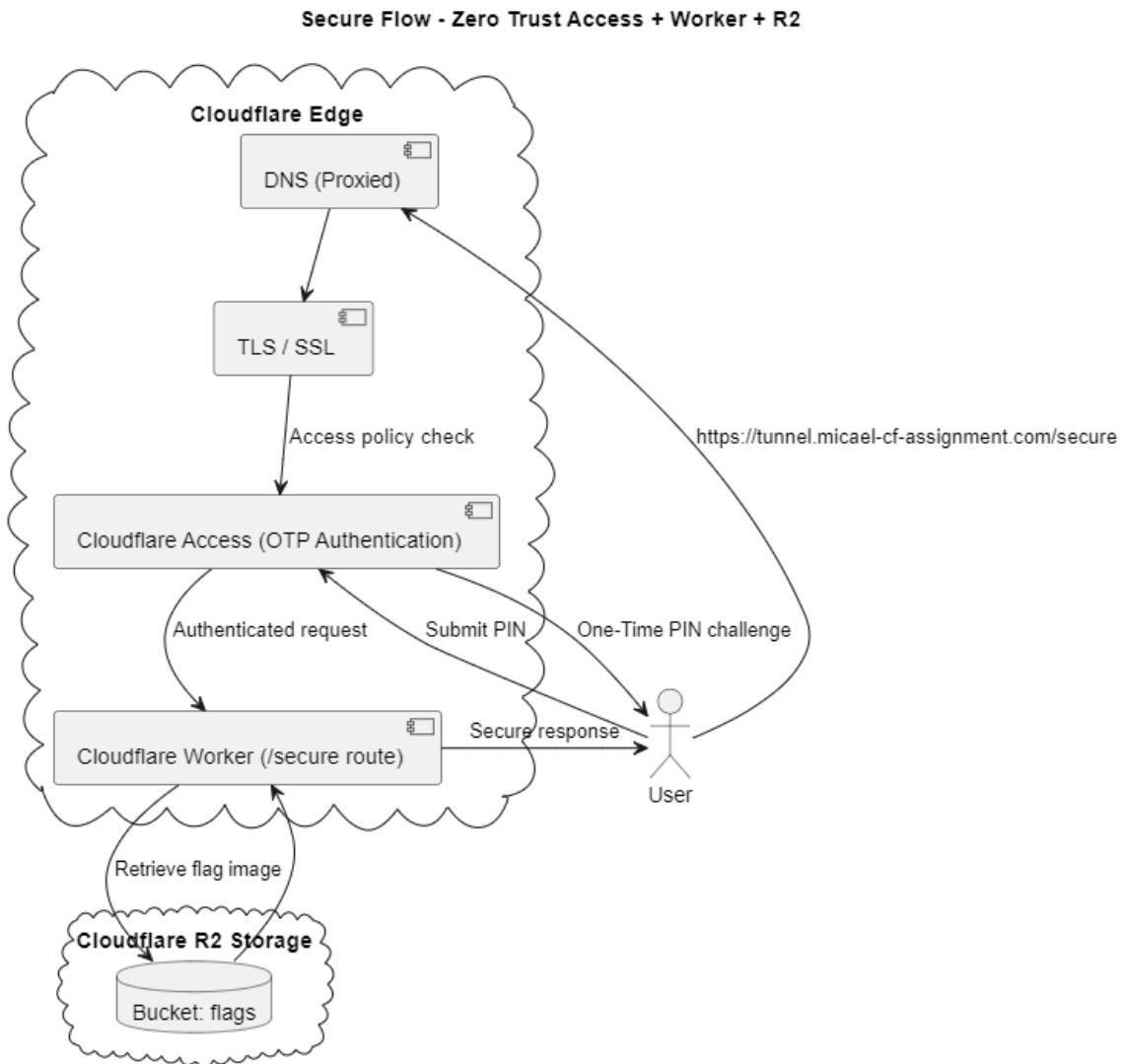


Fig. 3Secure diagram

4. Implementation Steps

Origin Web Server

An origin server was deployed on AWS EC2 using Ubuntu Linux.

A Flask application was created to return all HTTP request headers in the response body. The application was served using Gunicorn as a WSGI server and placed behind an NGINX reverse proxy.

The screenshot shows the AWS CloudWatch Instances console. At the top, there is a search bar and a filter section for 'Name' (with 'cf-Michael' selected). Below the search bar, a table lists one instance: 'cf-Michael' (Instance ID: i-03f2a72a48ad5501d), which is 'Running'. The instance type is 't3.micro', it has 3/3 checks passed, and its availability zone is 'eu-central-1b'. It has a Public IPv4 DNS of 'ec2-18-185-112-93.eu-central-1.compute.amazonaws.com' and a Public IPv4 IP of '18.185.112.93'. An 'Actions' dropdown menu is visible at the top right. Below the table, there is a detailed view for the instance 'i-03f2a72a48ad5501d (cf-Michael)'. The 'Details' tab is selected, showing various configuration details such as Instance ID (i-03f2a72a48ad5501d), Public IPv4 address (18.185.112.93), Instance state (Running), Private IP DNS name (ip-172-31-10-1.compute.internal), Instance type (t3.micro), VPC ID (vpc-0c12a...), and AWS Compute Optimizer finding (Opt-in to AWS Compute Optimizer for recommendations).

Fig. 4EC2 instance running

```
GNU nano 7.2
server {
    server_name micael-cf-assignment.com;

    location / {
        proxy_pass http://127.0.0.1:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/micael-cf-assignment.com/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/micael-cf-assignment.com/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    if ($host = micael-cf-assignment.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name micael-cf-assignment.com;
    return 404; # managed by Certbot
}
```

Fig. 5NGINX config file



Fig. 6Flask headers output

```
ubuntu@ip-172-31-38-51:~/headers-app$ systemctl status headers-app
● headers-app.service - Headers Flask App (Gunicorn)
   Loaded: loaded (/etc/systemd/system/headers-app.service; enabled; preset: enabled)
   Active: active (running) since Wed 2026-02-18 10:50:23 UTC; 1min 20s ago
     Main PID: 20637 (gunicorn)
        Tasks: 3 (limit: 1017)
       Memory: 29.3M (peak: 29.5M)
      CPU: 234ms
     CGroup: /system.slice/headers-app.service
             └─ 20637 /home/ubuntu/headers-app/venv/bin/python3 /home/ubuntu/headers-app/venv/bin/gunicorn -b 127.0.0.1:5000 wsgi:app
             ├─ 20643 /home/ubuntu/headers-app/venv/bin/python3 /home/ubuntu/headers-app/venv/bin/gunicorn -b 127.0.0.1:5000 wsgi:app

Feb 18 10:50:23 ip-172-31-38-51 systemd[1]: headers-app.service: Scheduled restart job, restart counter is at 9.
Feb 18 10:50:23 ip-172-31-38-51 systemd[1]: Started headers-app service - Headers Flask App (Gunicorn).
Feb 18 10:50:23 ip-172-31-38-51 gunicorn[20637] [2026-02-18 10:50:23 +0000] [20637] [INFO] Starting gunicorn 2.0.1
Feb 18 10:50:23 ip-172-31-38-51 gunicorn[20637] [2026-02-18 10:50:23 +0000] [20637] [INFO] Listening at: http://127.0.0.1:5000 (20637)
Feb 18 10:50:23 ip-172-31-38-51 gunicorn[20637] [2026-02-18 10:50:23 +0000] [20637] [INFO] Using worker: sync
Feb 18 10:50:23 ip-172-31-38-51 gunicorn[20637] [2026-02-18 10:50:23 +0000] [20637] [INFO] Control socket listening at /home/ubuntu/headers-app/gunicorn.ctl
Feb 18 10:50:23 ip-172-31-38-51 gunicorn[20643] [2026-02-18 10:50:23 +0000] [20643] [INFO] Booting worker with pid: 20643
ubuntu@ip-172-31-38-51:~/headers-app$ curl http://127.0.0.1:5000
Host: 127.0.0.1:5000
User-Agent: curl/8.5.0
Accept: */*
Host: 127.0.0.1:5000
User-Agent: curl/8.5.0
Accept: */*
Host: 127.0.0.1:5000
User-Agent: curl/8.5.0
Accept: */*
```

Fig. 7systemctl status headers-app

Proxy Through Cloudflare

The domain was onboarded into Cloudflare and DNS records were configured to proxy traffic through Cloudflare's network.

The orange-cloud proxy mode was enabled to ensure traffic passed through Cloudflare services.

micael-cf-assignment.com points to **18.185.112.93** and has its traffic proxied through Cloudflare.

Type	Name (required)	IPv4 address (required)	Proxy status	TTL	
A	@	18.185.112.93	Proxied	Auto	

Fig. 8Cloudflare DNS records

```
C:\Users\micael>curl -I http://micael-cf-assignment.com
HTTP/1.1 200 OK
Date: Tue, 17 Feb 2026 18:31:28 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Server: cloudflare
Nel: {"report_to": "cf-nel", "success_fraction": 0.0, "max_age": 604800}
cf-cache-status: DYNAMIC
Report-To: {"group": "cf-nel", "max_age": 604800, "endpoints": [{"url": "https://a.nel.cloudflare.com/report/v4?s=rjfL58ACX%2F61kjdsEhuS6WG8tYAnaNfwMnYKFVb1c8C6SIiyfUVMH9sbEYHQkESJFNDbwSaocEjd7kM6neNDAb9n5BUAwsFHEV2zvwu%2BQ4LLFRzIYs8UW%2Fk0tBrz7PpcbRYzA%3D%3D"}]}
CF-RAY: 9cf75ca14eb534bb-BRU
alt-svc: h3=":443"; ma=86400
```

Fig. 9curl showing Cloudflare headers

Full Strict TLS

A TLS certificate was generated on the origin server using Let's Encrypt.

Cloudflare SSL mode was configured to Full (Strict), ensuring encrypted communication between Cloudflare and the origin server with certificate validation.

```

Account registered.
Requesting a certificate for micael-cf-assignment.com

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/micael-cf-assignment.com/fullchain.pem
Key is saved at:          /etc/letsencrypt/live/micael-cf-assignment.com/privkey.pem
This certificate expires on 2026-05-18.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

Deploying certificate
Successfully deployed certificate for micael-cf-assignment.com to /etc/nginx/sites-enabled/default
Congratulations! You have successfully enabled HTTPS on https://micael-cf-assignment.com

-----[REDACTED]-----
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
 * Donating to EFF:                  https://eff.org/donate-le
-----[REDACTED]-----[REDACTED]
ubuntu@ip-172-31-38-51:/$ █

```

Fig. 10App certificate Let's Encrypt

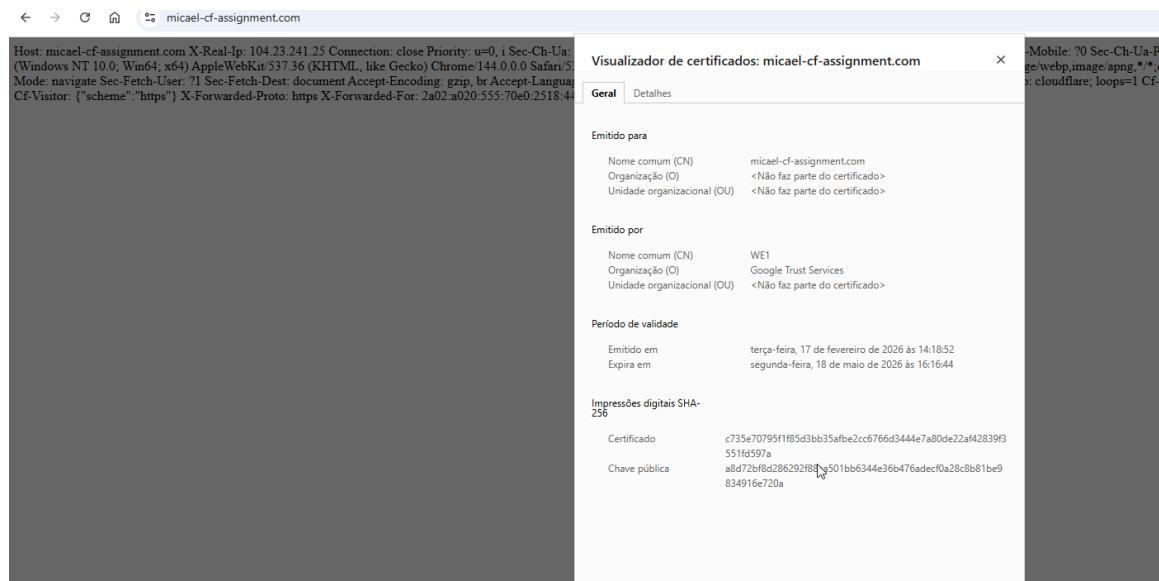


Fig. 11Certificate details browser (Cloudflare)

Micael.tavares.santo... ⚡

micael-cf-assignment.com ⚡ Free

Quick search... Ctrl K

Back to Domains

Overview

Recents

AI Crawl Control

Log Explorer

Analytics & logs

DNS

Email

SSL/TLS

Overview

Edge Certificates

Client Certificates

Origin Server

Custom Hostnames

Security

Access

Speed

Caching

Workers Routes

Rules

Error Pages New

Network

Select encryption mode

Automatic SSL/TLS (default)

Cloudflare checks if your traffic needs a more secure encryption mode and updates your setting automatically.

Select

Custom SSL/TLS

Select the encryption mode that Cloudflare uses to connect to your origin server

Selected

Full (Strict)

Enable encryption end-to-end and enforce validation on origin certificates. Use Cloudflare's Origin CA to generate certificates for your origin.

Full

Enable encryption end-to-end. Use this mode when your origin server supports SSL certification but does not use a valid, publicly trusted certificate.

Flexible

Enable encryption only between your visitors and Cloudflare. This will avoid browser security warnings, but all connections between Cloudflare and your origin are made through HTTP.

Off (not secure)

No encryption applied. Turning off SSL disables HTTPS and causes browsers to show a warning that your website is not secure.

The diagram illustrates the SSL/TLS connection flow. It starts with a 'Browser' icon on the left, which has a blue arrow pointing to a 'Cloudflare' icon in the middle. The 'Cloudflare' icon is orange with a white cloud and a lock symbol, and the word 'Cloudflare' is written below it. A second blue arrow points from the 'Cloudflare' icon to an 'Origin Server' icon on the right. The 'Origin Server' icon features a computer monitor, a padlock, and a magnifying glass.

Cancel Save

Fig. 12Cloudflare SSL settings page

Cloudflare Tunnel

Cloudflare Tunnel was installed and configured on the origin server to securely expose the application without opening inbound firewall ports.

A subdomain tunnel.mydomain.com was configured to route traffic to the local NGINX service.

```
ubuntu@ip-172-31-38-51:~$ cloudflared tunnel login
Please open the following URL and log in with your Cloudflare account:
https://dash.cloudflare.com)argotunnel?aud=&callback=https%3A%2F%2Flogin.cloudflareaccess.org%2FyeiVTR702CDgGIek1zIxrOu08QxA8ruif_F_E42ha%3D

Leave cloudflared running to download the cert automatically.
2026-02-17T20:24:27Z  INFO Waiting for login...
2026-02-17T20:25:20Z  INFO Waiting for[login...
2026-02-17T20:25:21Z  INFO You have successfully logged in.
If you wish to copy your credentials to a server, they have been saved to:
/home/ubuntu/.cloudflared/cert.pem
```

Fig. 13 cloudflared running

```
ubuntu@ip-172-31-36-51: ~$ sudo cloudflare tunnel -config /etc/cloudflare/cf-tunnel.json run cf-tunnel
INFO Starting tunnel tunnel@ip-172-31-36-51:~$cf08-cfbb-4de4-a10-ea1c95e4fb6
[02-04-02-172-31-36-51] INFO Version: 2026.2.0 (Checksum:0x767464bdbe7bd74d8ff3d287e8930e4645ebbe6700f803fddda5a4c307c16)
[02-04-02-172-31-36-51] INFO GOOS: linux, GOVersion: 0x24.13, GoArch: amd64
[02-04-02-172-31-36-51] INFO settings: map[config:/etc/cloudflare/config:cf-tunnel.cred-file:/home/ubuntu/.cloudflared/c69ccf08-cf8fb-4de4-a10-ea1c95e4f60b.json credentials-file:/home/ubuntu/.cloudflared/c69ccf08-cf8fb-4de4-a10-ea1c95e4f60b.json]
[02-04-02-172-31-36-51] INFO cloudflared will not automatically update when run from the shell. To enable auto-updates, run cloudflared as a service: https://developers.cloudflare.com/cloudflare-one/connections/connect-apps/configure-tunnel/enable-automatic-updates-as-a-service/
[02-04-02-172-31-36-51] INFO Using default connection configuration: 0x8068edc-fa18-402f-a0cb-2f9d57cfae2
[02-04-02-172-31-36-51] INFO Initial protocol quit
[02-04-02-172-31-36-51] INFO ICMP proxy with ip 172.31.38.51 as source for IPv4
[02-04-02-172-31-36-51] INFO ICMP proxy with ip 172.31.38.51 as source for IPv6
[02-04-02-172-31-36-51] INFO The user running cloudflared process has a GID (group ID) that is not within ping_group_range. You might need to add that user to a group within that range, or instead update the range to encompass a group the user is already in by modifying /proc/sys/net/ipv4/ping_group_range. Otherwise cloudflared will not be able to ping this network error="Group ID 0 is not between ping group 1 to 0"
[02-04-02-172-31-36-51] INFO ICMP proxy feature is disabled
[02-04-02-172-31-36-51] INFO ICMP proxy with ip 172.31.38.51 as source for IPv4
[02-04-02-172-31-36-51] INFO ICMP proxy with ip 172.31.38.51 as source for IPv6
[02-04-02-172-31-36-51] INFO Starting metrics server on 127.0.0.1:20241/metrics
[02-04-02-172-31-36-51] INFO Tunnel connection curve preferences: [x2519mgkem761 curveIndex=0] connIndex=0 event=0 ip=198.41.192.167
[02-04-02-172-31-36-51] INFO Registered tunnel connection connIndex=0 connection=x2519mgkem761 curveIndex=0 event=0 ip=198.41.192.167 location=fra10 protocol=quic
[02-04-02-172-31-36-51] INFO Tunnel connection curve preferences: [x2519mgkem761 curveIndex=1] connIndex=1 event=0 ip=198.41.200.113
[02-04-02-172-31-36-51] INFO Registered tunnel connection connIndex=1 connection=x2519mgkem761 curveIndex=1 event=0 ip=198.41.200.113 location=fra10 protocol=quic
[02-04-02-172-31-36-51] INFO Tunnel connection curve preferences: [x2519mgkem761 curveIndex=2] connIndex=2 event=0 ip=198.41.192.7
[02-04-02-172-31-36-51] INFO Registered tunnel connection connIndex=2 connection=x2519mgkem761 curveIndex=2 event=0 ip=198.41.192.7 location=fra10 protocol=quic
[02-04-02-172-31-36-51] INFO Tunnel connection curve preferences: [x2519mgkem761 curveIndex=3] connIndex=3 event=0 ip=198.41.200.53
[02-04-02-172-31-36-51] INFO Registered tunnel connection connIndex=3 connection=x2519mgkem761 curveIndex=3 event=0 ip=198.41.200.53 location=fra10 protocol=quic
```

Fig. 14 Tunnel running

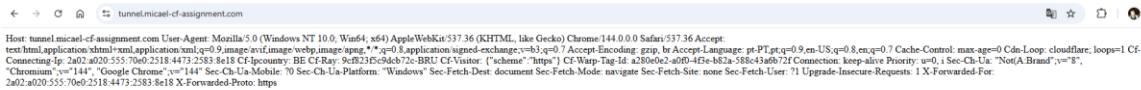


Fig. 15 Browser accessing tunnel URL

Zero Trust IdP

Cloudflare Zero Trust Access was configured using One-Time PIN (OTP) authentication as the Identity Provider.

This allowed secure user authentication without integrating an external identity platform.

Name	Identity provider	Test
One-time PIN	One-time PIN	

Fig. 16 IdP OTP

Fig. 17 App config for OTP

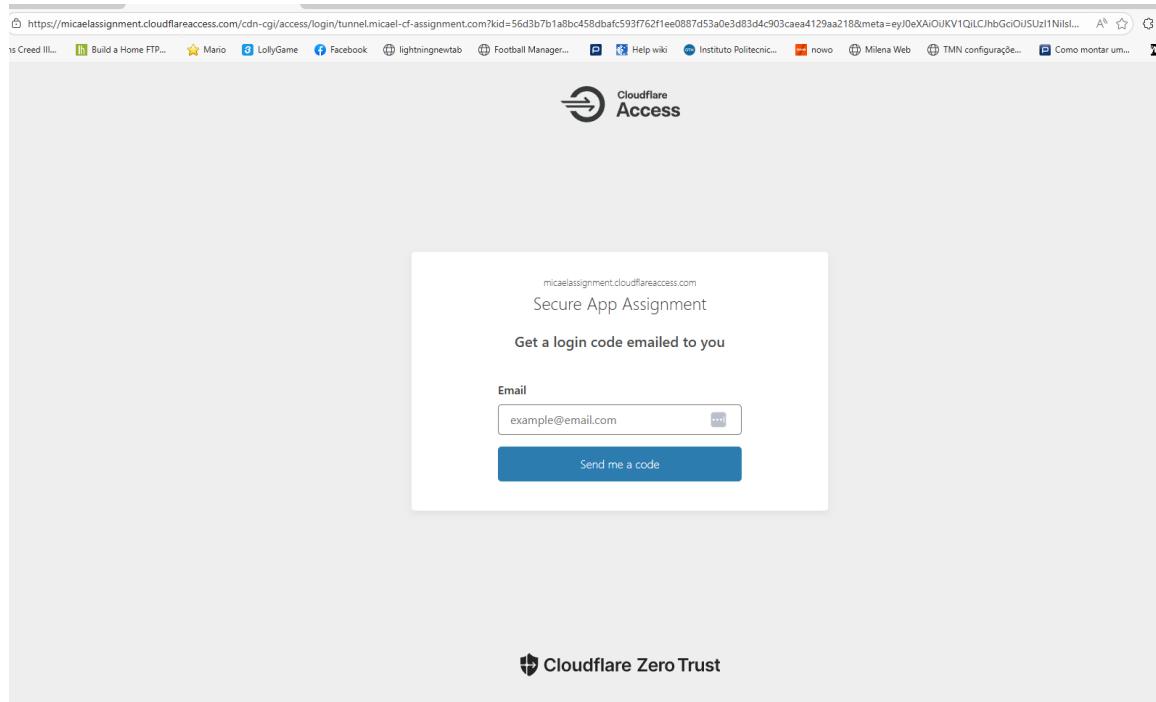


Fig. 18 Login screen

Secure Path Protection

Access policies were configured to protect the /secure path of the tunnel subdomain.

Access was restricted to:

- The project owner email
- Users with @cloudflare.com domain

Direct access to the origin IP was prevented by security group configuration and by routing traffic exclusively through Cloudflare Tunnel.

[← Back to Applications](#)

Add an application

Configure the policies, authentication, and settings of your application.

Select type > **Configure application** > Experience settings (optional) > Advanced settings (optional)

Basic information

Configure your application's basic details and paths. Enter hostnames or IPs to protect an entire website or specific subdomains and paths.

Application name (Required)

Assignment Secure App

Session Duration (Required)

1 month

Public hostname

Input Subdomain

method

D... ▾

tunnel

Domain (Required)

micael-cf-assignment.com ▾

X

Path

/

secure*

+ Add public hostname

+ Add private hostname

+ Add private IP

Fig. 19 Access Policy rule

Include OR
If more than one Include rule is configured, users only need to meet one of the criteria.

Selector (Required)	Value
Emails	micael.tavares.santos@gmail.com ✎ email@example.com

Selector (Required)	Value
Emails ending in	@cloudflare ✎

+ Add include + Add require + Add exclude

Policy tester
The policy tester evaluates the last seen identity of active users. Login decisions may differ if there are changes to user attributes evaluated by this policy.

[Test policy](#)

1 user (100%) is blocked

Username	Email	Status
micaelsantos_69	micaelsantos_69@hotmail.com	BLOCKED

1 - 1 Items per page: 5 < 1 of 1 page >

Fig. 20 Block test

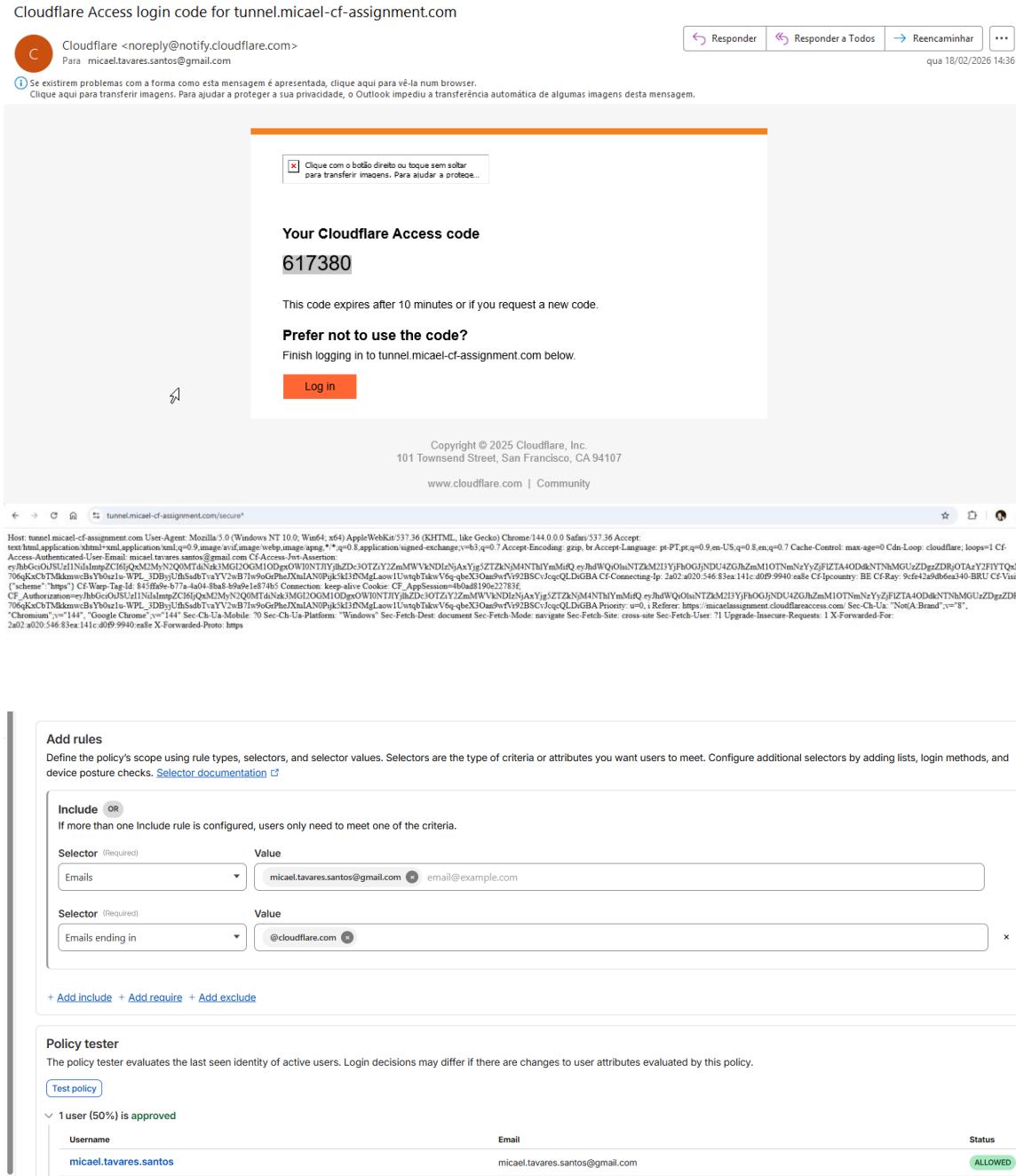


Fig. 21 Working Test

Worker + R2

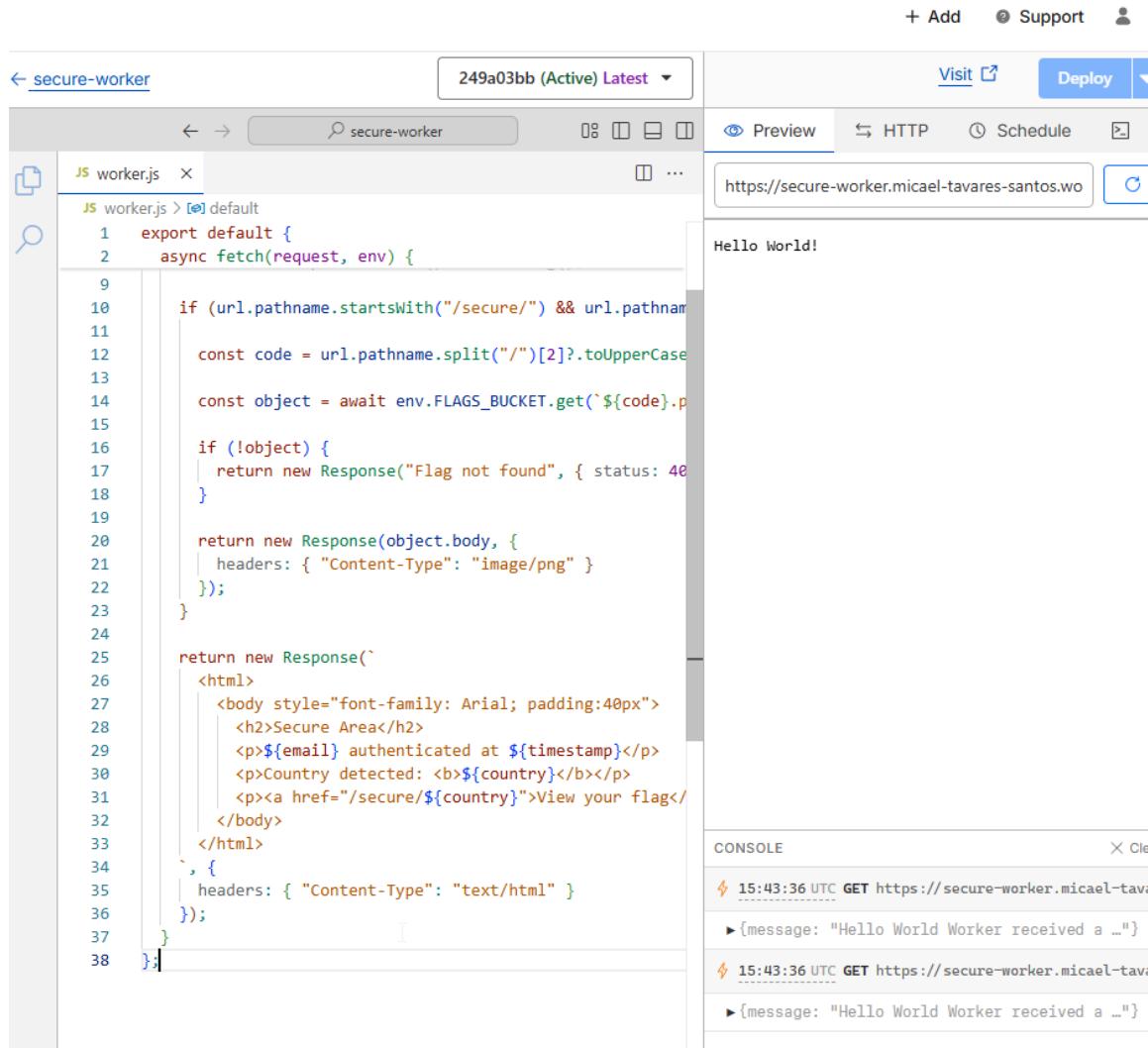
A Cloudflare Worker was created to run on the /secure path.

The Worker:

- Extracts user identity information from Cloudflare Access headers
 - Displays authentication metadata

- Retrieves country flag images from a private R2 bucket

The Worker was deployed using the Wrangler CLI and code was uploaded to a public Git repository.



The screenshot shows the Cloudflare Workers dashboard for a worker named "secure-worker".

- Code View:** The "worker.js" file contains the following code:

```

1  export default {
2    async fetch(request, env) {
3
4      if (url.pathname.startsWith("/secure/") && url.pathname
5          const code = url.pathname.split("/")[2]?_.toUpperCase()
6
7          const object = await env.FLAGS_BUCKET.get(`#${code}.p
8
9          if (!object) {
10            return new Response("Flag not found", { status: 404 })
11
12            return new Response(object.body, {
13              headers: { "Content-Type": "image/png" }
14            });
15
16
17            return new Response(`<html>
18              <body style="font-family: Arial; padding:40px">
19                <h2>Secure Area</h2>
20                <p>${email} authenticated at ${timestamp}</p>
21                <p>Country detected: <b>${country}</b></p>
22                <p><a href="/secure/${country}">View your flag</a>
23              </body>
24            </html>
25            , {
26              headers: { "Content-Type": "text/html" }
27            });
28
29
30
31
32
33
34
35
36
37
38

```
- Preview:** The preview URL is <https://secure-worker.micael-tavares-santos.wo>, and the output is "Hello World!"
- Logs:** The console log shows two entries:
 - 15:43:36 UTC GET https://secure-worker.micael-tavares-santos.wo
 - ▶ {message: "Hello World Worker received a ..."} (repeated twice)

Fig. 22Worker code

R2 Object Storage > flags

+ Add Support

Default Storage Class ⓘ	Public Access ⓘ	Bucket Size	Class A Operations ⓘ	Class B Operations ⓘ
Standard	Disabled	0 B	10	30

Objects Metrics Settings

Search objects by prefix

 View prefixes as directories ⓘ

flags /

Upload + Add directory

<input type="checkbox"/> Objects	Type	Storage Class	Size	Modified
<input type="checkbox"/> BE.png	image/png	Standard	292 B	18 Feb 2...
<input type="checkbox"/> DE.png	image/png	Standard	151 B	18 Feb 2...
<input type="checkbox"/> ES.png	image/png	Standard	3.28 ...	18 Feb 2...
<input type="checkbox"/> FR.png	image/png	Standard	254 B	18 Feb 2...
<input type="checkbox"/> IT.png	image/png	Standard	253 B	18 Feb 2...
<input type="checkbox"/> LU.png	image/png	Standard	151 B	18 Feb 2...

Drag and drop to start uploading

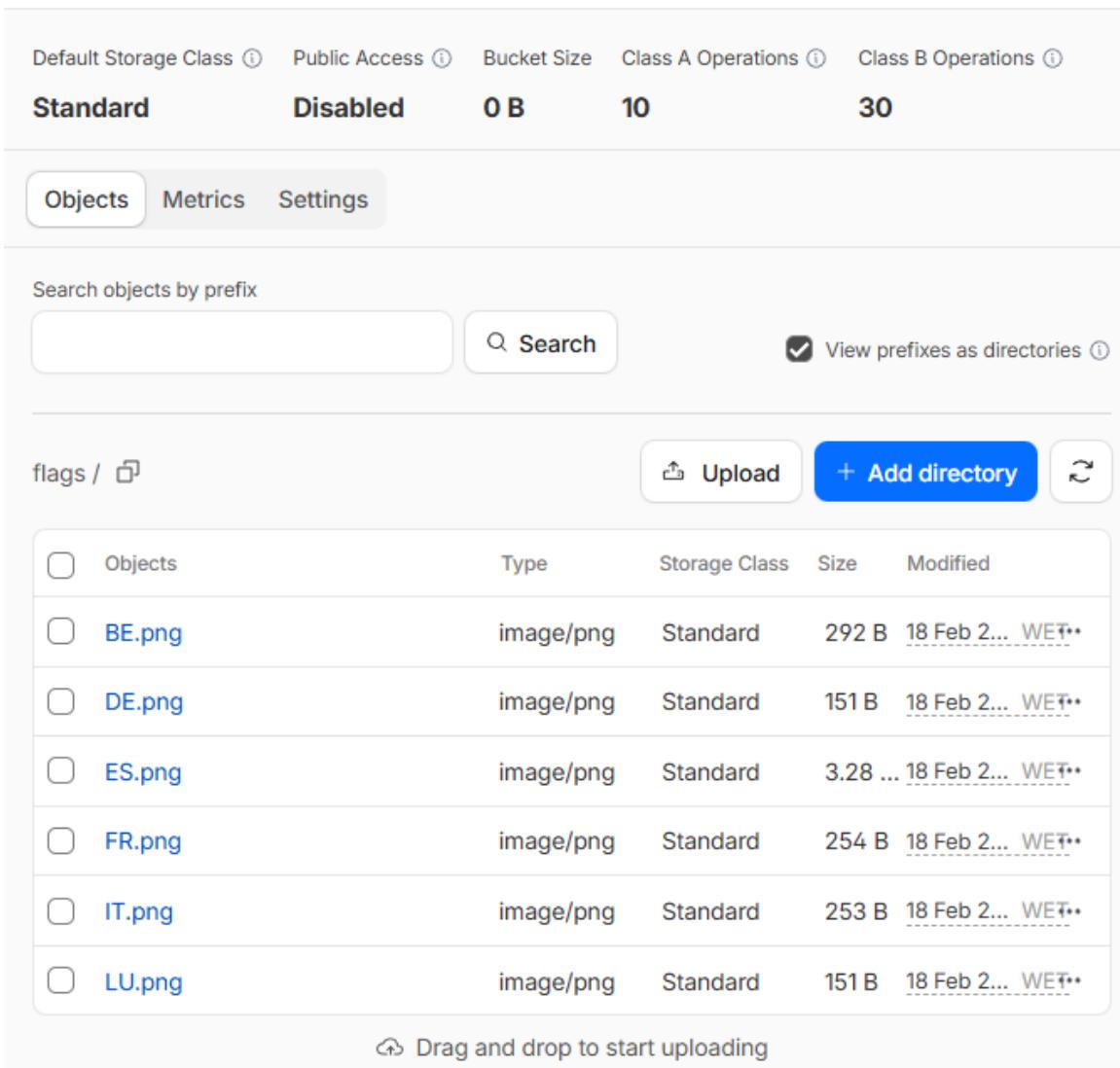


Fig. 23R2 bucket

Workers & Pages > secure-worker

+ Add Support

Metrics Deployments Bindings Observability Settings

Edit code Visit

Bindings

Add and connect external resources to your Worker without needing to manage permissions or API keys.

[View docs](#) [Add binding +](#)

Connected Bindings

The diagram illustrates a binding between a worker and an R2 bucket. A central node labeled "secure-worker" is connected by a line to a blue rounded rectangle labeled "R2 bucket". This rectangle contains the text "FLAGS_BUCKET" and a small "Binding" label below it. The entire interface is set against a light gray dotted background.

Type	Name	Value	Actions
R2 bucket	FLAGS_BUCKET	<u>flags</u>	

Fig. 24 Worker binding



Fig. 25Secure page working

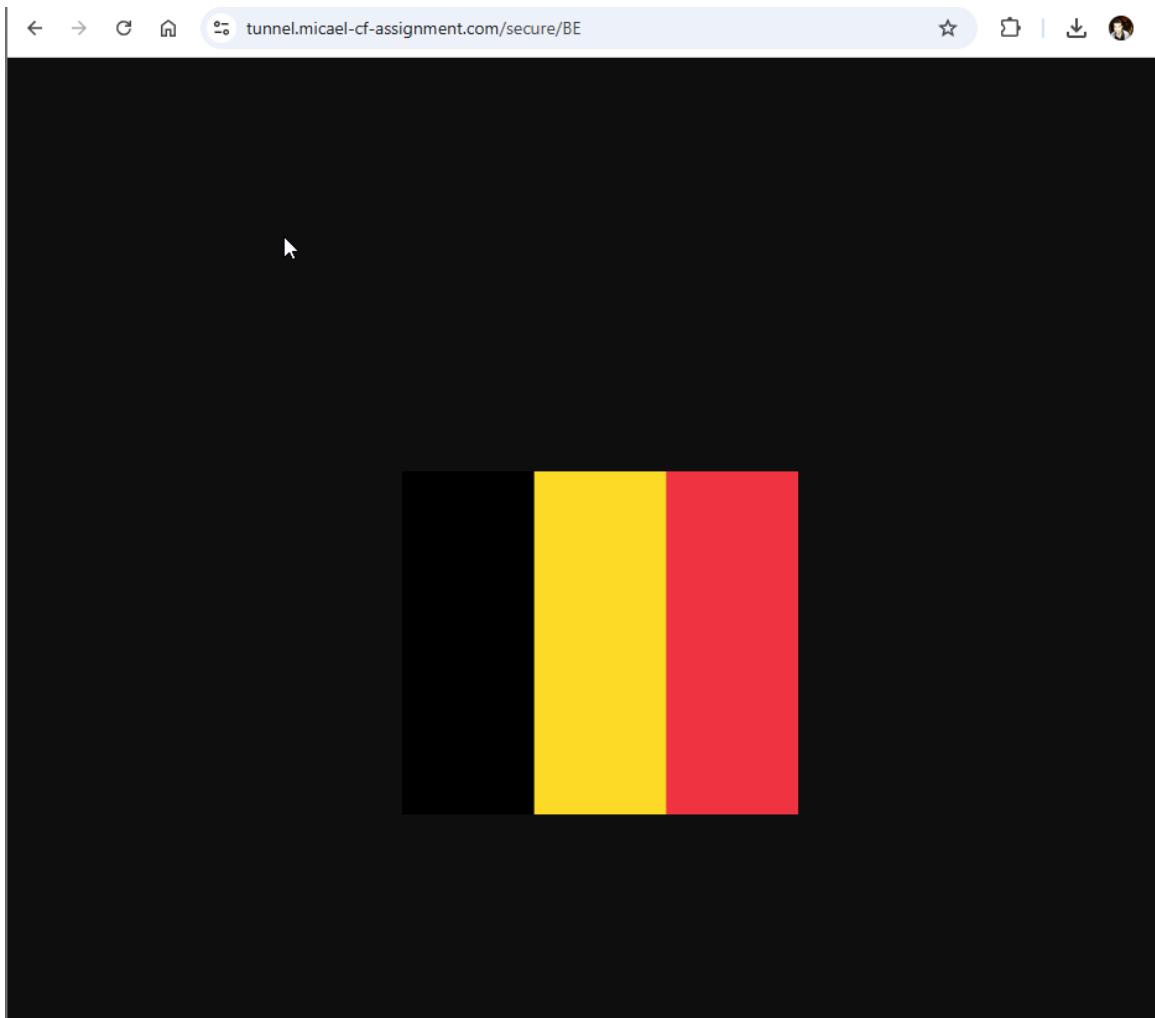


Fig. 26Flag displayed

5. Security Controls Implemented

The following security controls were implemented:

- TLS encryption end-to-end
- Full Strict certificate validation
- Cloudflare Tunnel to eliminate exposed ports
- Zero Trust authentication policies
- Identity-based access control
- Private object storage with R2
- Reverse proxy isolation
- Origin firewall restrictions

6. Use Cases

The implemented architecture is relevant for multiple enterprise scenarios:

- Secure remote access to internal applications
- Zero Trust replacement for VPN access
- Protection of legacy applications without modification
- Identity-aware content delivery
- Secure asset delivery using object storage
- Dev/Test environment exposure without public IP risks

7. Knowledge Gaps and Research

During the implementation process, several knowledge gaps were identified and addressed through research and experimentation.

Examples include:

- Cloudflare Tunnel configuration and troubleshooting
- Systemd service configuration for persistent application execution
- Cloudflare Zero Trust Access policy behaviour
- Worker and R2 integration
- SSL Full Strict troubleshooting

Primary research sources included:

- Official Cloudflare documentation
- Community technical resources
- Hands-on testing and validation
- AI-assisted guidance for development efficiency

AI tools were used specifically to accelerate:

- Code generation (Flask and Worker examples)
- Troubleshooting ideas
- Documentation structure guidance

All configurations were validated manually to ensure correctness.

8. Customer Experience Perspective

From a customer perspective, the experience would be highly positive.

Cloudflare provides a unified platform to secure applications without requiring significant infrastructure changes.

The ability to deploy Zero Trust controls, secure tunnels and edge compute capabilities rapidly demonstrates strong value, particularly for organizations seeking to modernize legacy environments or reduce VPN dependency.

The learning curve is moderate, but documentation quality and platform integration significantly reduce operational complexity.

9. Conclusion

This project successfully demonstrates the deployment of a secure application architecture using Cloudflare Application Services.

The implementation shows practical understanding of Cloudflare's core capabilities including secure connectivity, identity-based access control and edge computing.

The architecture is scalable, secure and aligned with modern Zero Trust principles.

10. Appendix (GitHub + URLs)

Public Application:

<https://tunnel.micael-cf-assignment.com>

Secure Application:

<https://tunnel.micael-cf-assignment.com/secure>

GitHub Repository:

<https://github.com/MicaelTavaresSantos/cloudflare-assignment-Micael.git>