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.

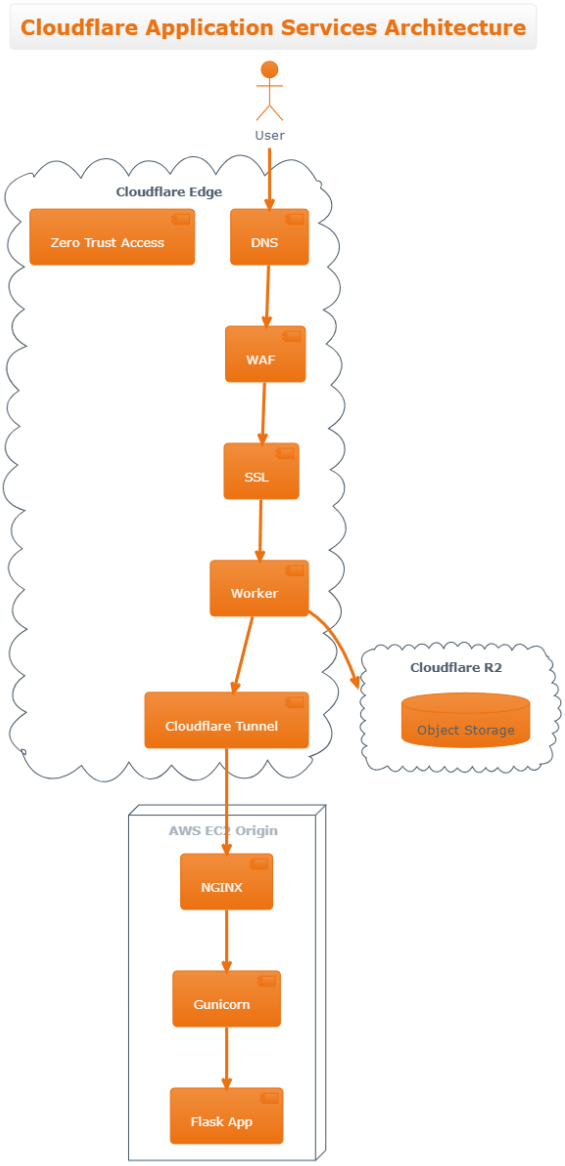


Fig. 1Architecture diagram

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

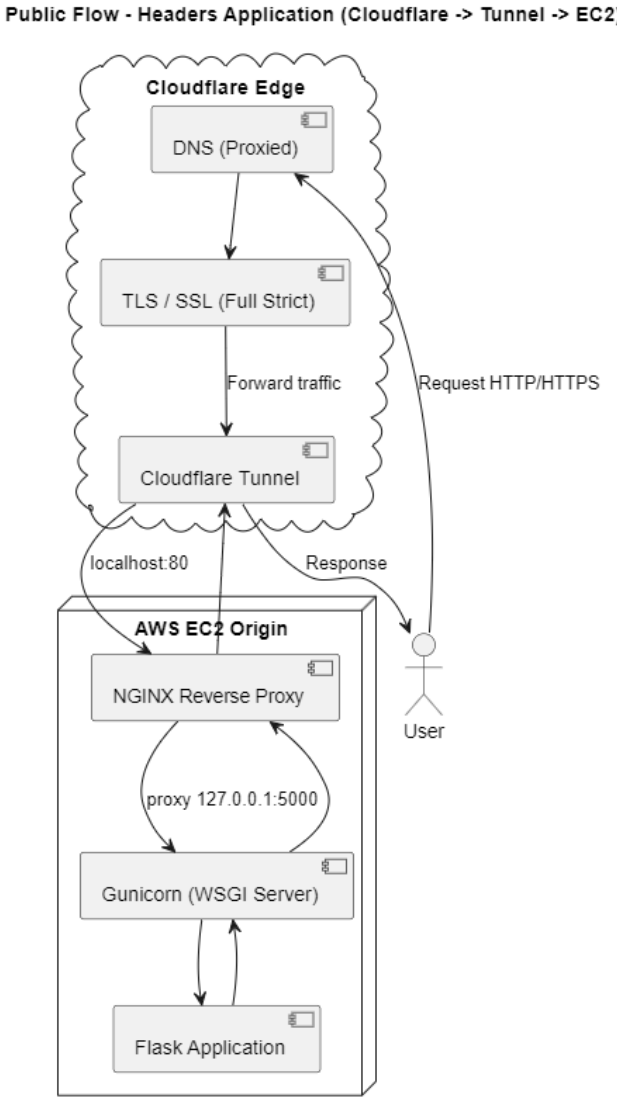


Fig. 2Public diagram

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.

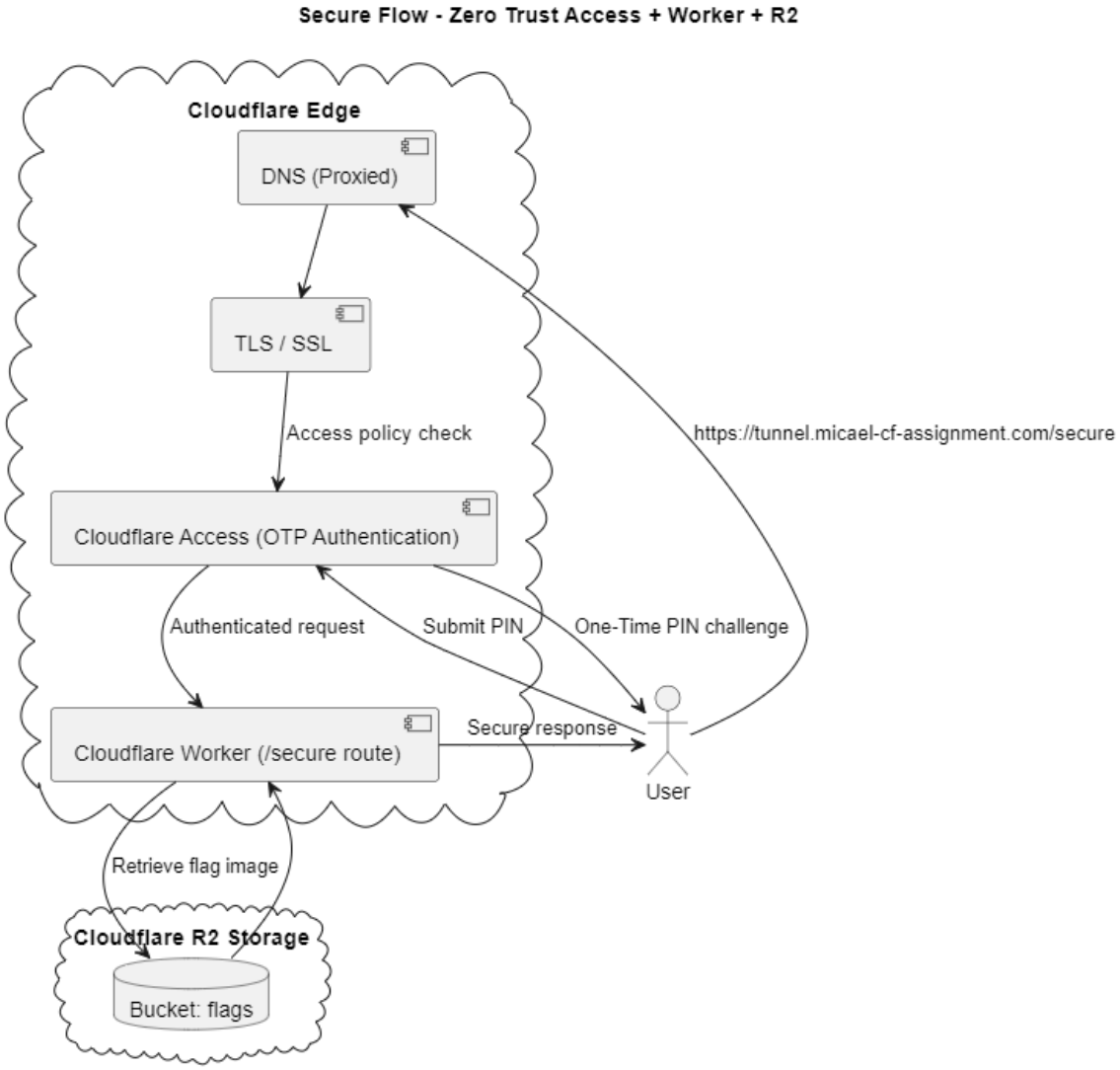


Fig. 3Secure diagram

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.

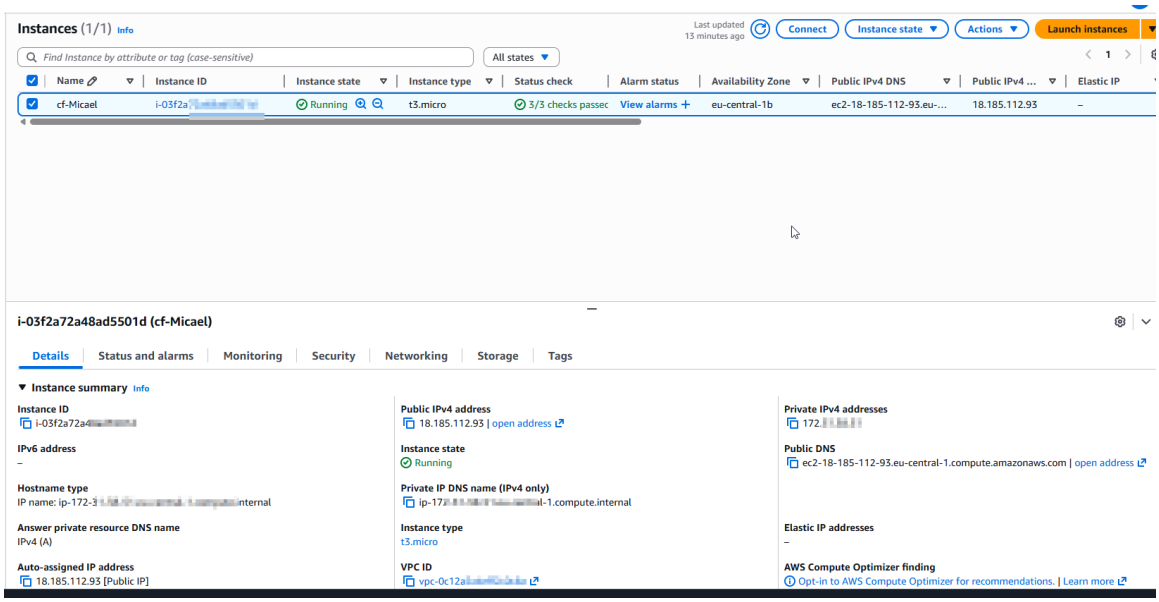


Fig. 4EC2 instance running

```
 GNU nano 7.2                                    /etc/nginx/sites-available/default
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

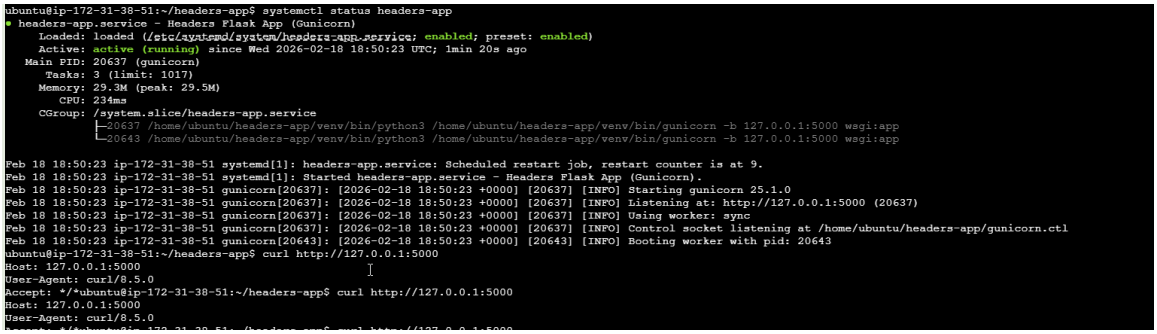


Fig. 7systemctl status headers-app

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.

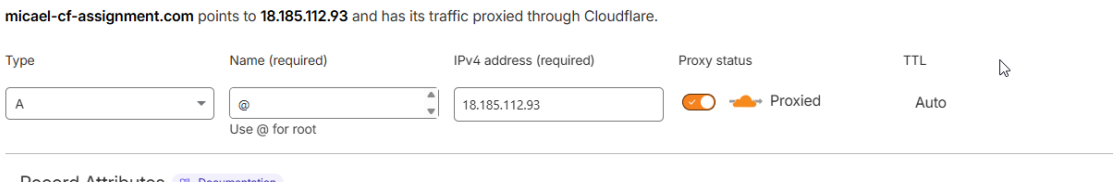


Fig. 8Cloudflare DNS records

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.

Fig. 10App certificate Let's Encrypt

Fig. 11Certificate details browser (Cloudflare)

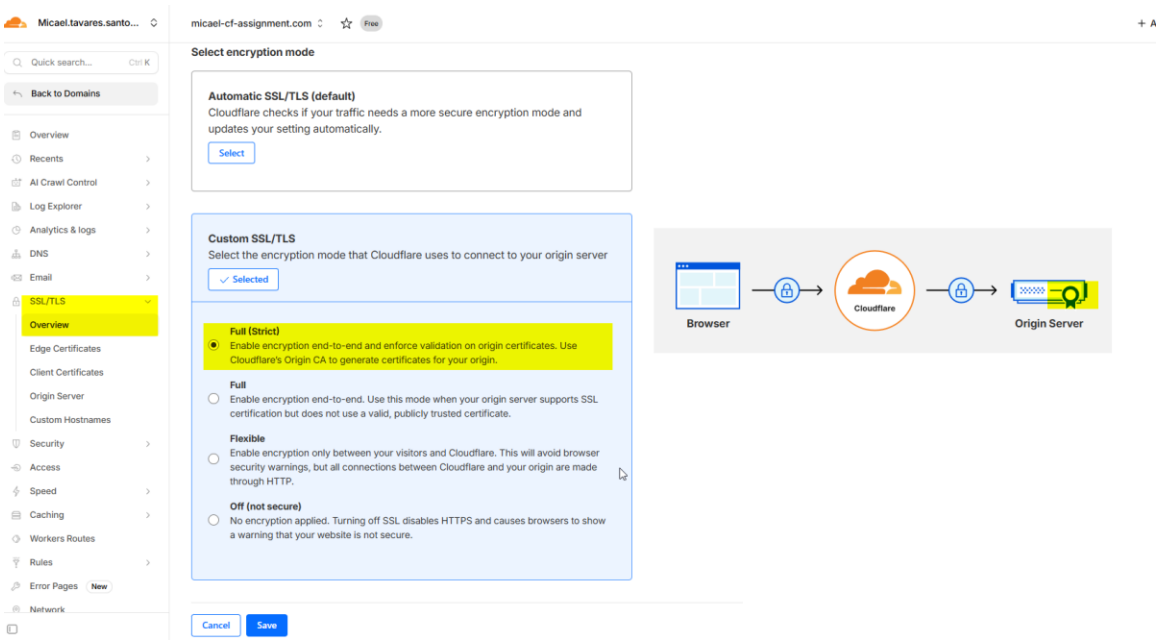


Fig. 12Cloudflare SSL settings page

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%2FyeiVTR702CDgGIekI5zIDx1zrOuO8QxASruiF_E42hs%3D

Leave cloudflared running to download the cert automatically.
2026-02-17T20:24:27Z INF Waiting for login...
2026-02-17T20:25:20Z INF Waiting for login...
2026-02-17T20:25:21Z INF 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. 13cloudflared running

```
ubuntu@ip-172-31-38-51:/$ sudo cloudflared tunnel --config /etc/cloudflared/config.yml run cf-tunnel
2026-02-17T20:40:43Z INF Starting tunnel tunnelID=c69ccf08-c8fb-4de6-a100-ea1c59e4f60b
2026-02-17T20:40:43Z INF Version 2026.2.0 (Checksum 176746db3be7dc7bd48f3dd287c8930a4645ebb6e6700f883fddda5a4c307c16)
2026-02-17T20:40:43Z INF GOOS: linux, GOVersion: go1.24.13, GoArch: amd64
2026-02-17T20:40:43Z INF Settings: map[config:/etc/cloudflared/config.yml cred-file:/home/ubuntu/.cloudflared/c69ccf08-c8fb-4de6-a100-ea1c59e4f60b.json credentials-file:/home/ubuntu/.cloudflared/c69ccf08-c8fb-4de6-a100-ea1c59e4f60b.json]
2026-02-17T20:40:43Z INF 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-tunnels/local-management/as-a-service/
2026-02-17T20:40:43Z INF Generated Connector ID: 0b806de5-fa18-402f-a0cb-2f9d57e4ase2
2026-02-17T20:40:43Z INF Initial protocol quic
2026-02-17T20:40:43Z INF ICMP proxy will use 172.31.38.51 as source for IPv4
2026-02-17T20:40:43Z INF ICMP proxy will use fe80::42e:6dff:fe41:8d89 in zone ens5 as source for IPv6
2026-02-17T20:40:43Z WRN 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" 
2026-02-17T20:40:43Z WRN ICMP proxy feature is disabled error="cannot create ICMPv4 proxy: Group ID 0 is not between ping group 1 to 0 nor ICMPv4 proxy: socket: permission denied"
2026-02-17T20:40:43Z INF ICMP proxy will use 172.31.38.51 as source for IPv4
2026-02-17T20:40:43Z INF ICMP proxy will use fe80::42e:6dff:fe41:8d89 in zone ens5 as source for IPv6
2026-02-17T20:40:43Z INF Starting metrics server on 127.0.0.1:20241/metrics
2026-02-17T20:40:44Z INF Tunnel connection curve preferences: [X25519MLKEM768 CurveP256] connIndex=0 event=0 ip=198.41.192.167
2026-02-17T20:40:44Z INF Registered tunnel connection connIndex=0 connection=84dafa72-6b1a-41e9-ac9b-4f979bfbbb44 event=0 ip=198.41.192.167 location=fra10 protocol=quic
2026-02-17T20:40:44Z INF Tunnel connection curve preferences: [X25519MLKEM768 CurveP256] connIndex=1 event=0 ip=198.41.200.113
2026-02-17T20:40:45Z INF Tunnel connection curve preferences: [X25519MLKEM768 CurveP256] connIndex=2 event=0 ip=198.41.192.7
2026-02-17T20:40:45Z INF Registered tunnel connection connIndex=2 connection=54765e55-65b9-46fd-beba-54879079cf4b event=0 ip=198.41.192.7 location=fra08 protocol=quic
2026-02-17T20:40:45Z INF Registered tunnel connection connIndex=1 connection=a115e5ca-9946-40fc-98ed-aa5d1db2e4ca event=0 ip=198.41.200.113 location=fra20 protocol=quic
2026-02-17T20:40:46Z INF Tunnel connection curve preferences: [X25519MLKEM768 CurveP256] connIndex=3 event=0 ip=198.41.200.53
2026-02-17T20:40:47Z INF Registered tunnel connection connIndex=3 connection=cad60b65-4f4d-4af1-a8ce-0df013f90874 event=0 ip=198.41.200.53 location=fra20 protocol=quic
```

Fig. 14Tunnel running

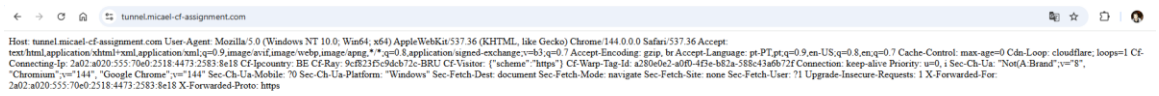


Fig. 15Browser accessing tunnel URL

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.

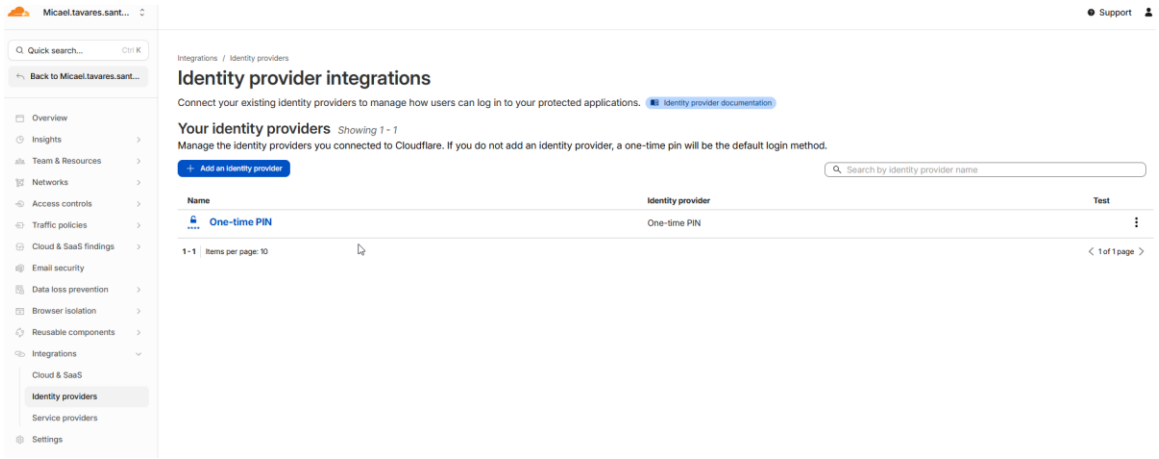


Fig. 16IdP OTP

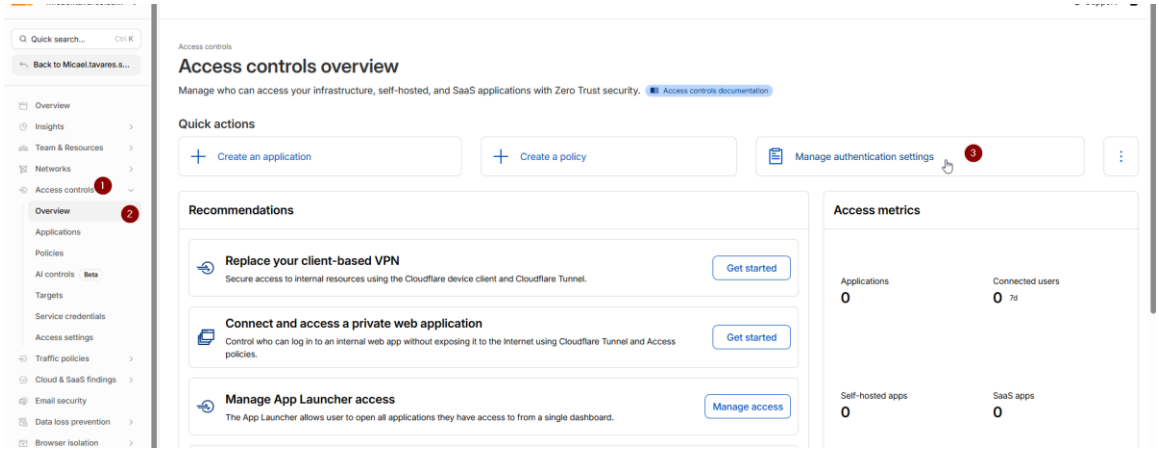


Fig. 17App config for OTP

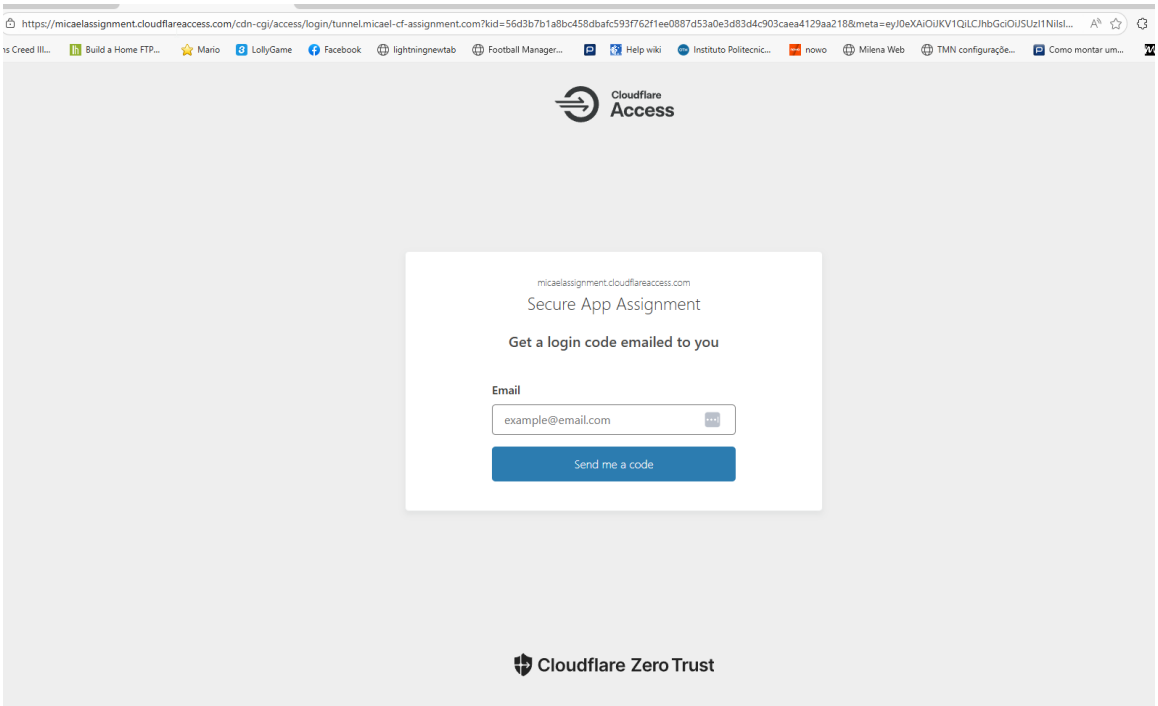


Fig. 18Login screen

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.

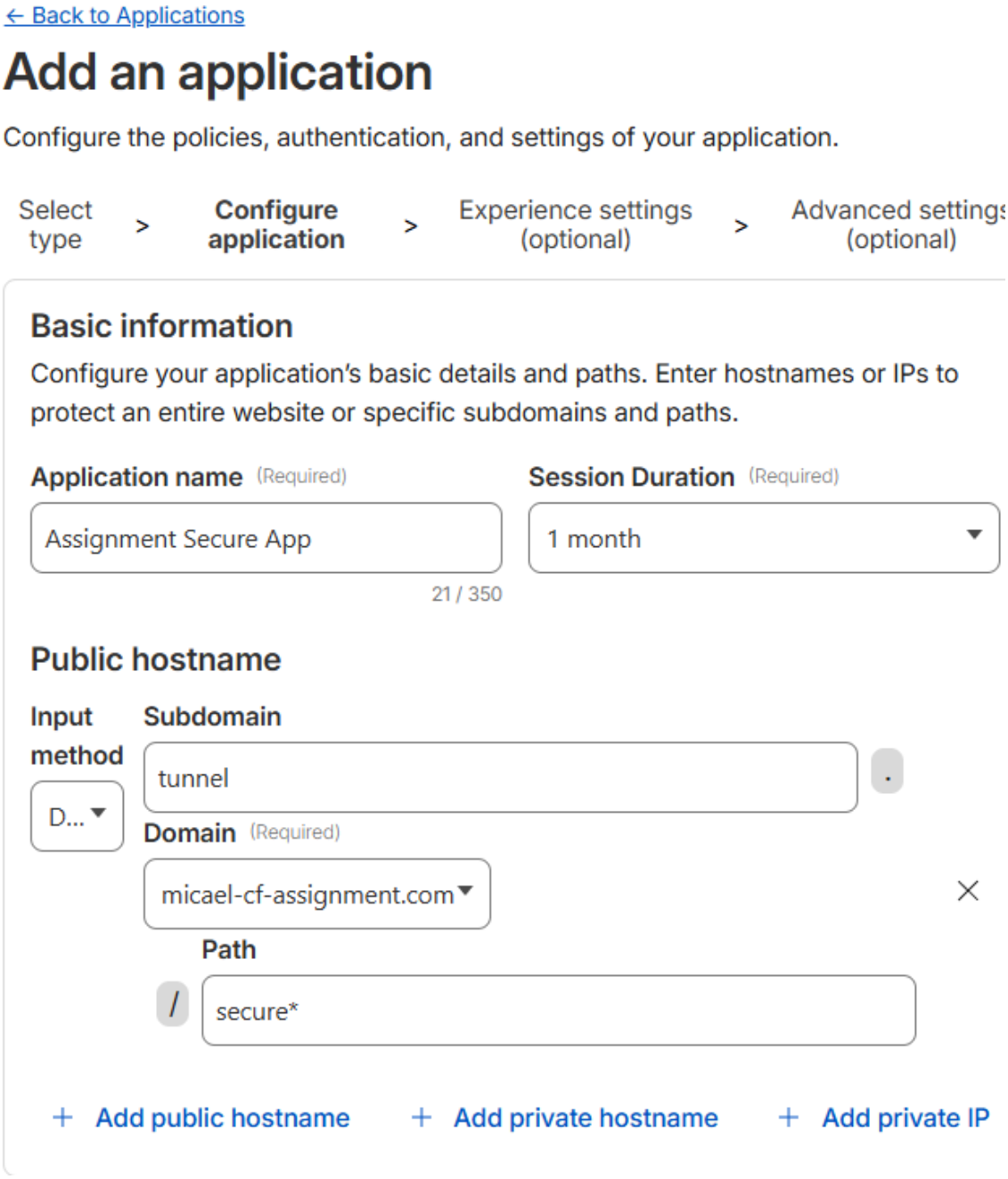


Fig. 19Access Policy rule

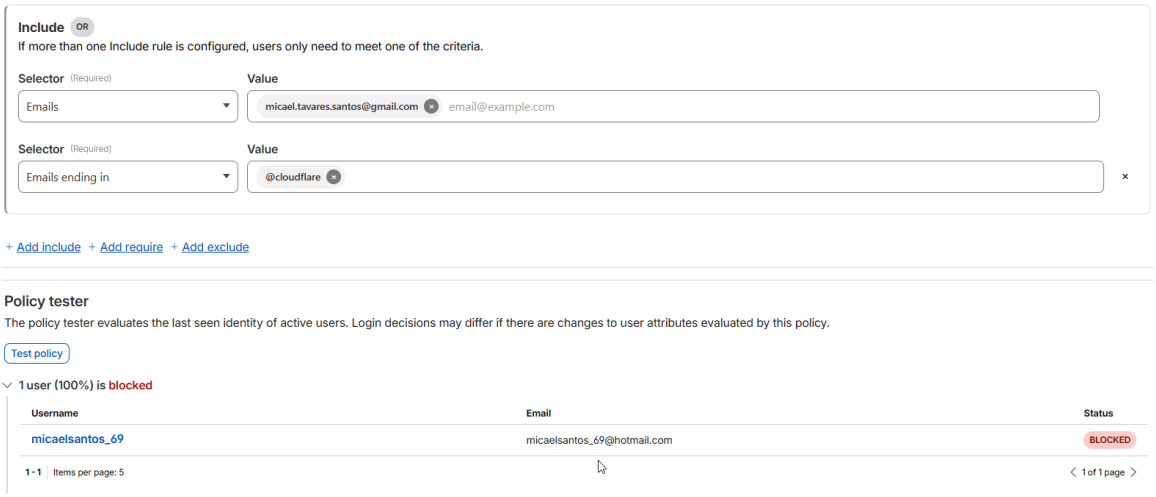
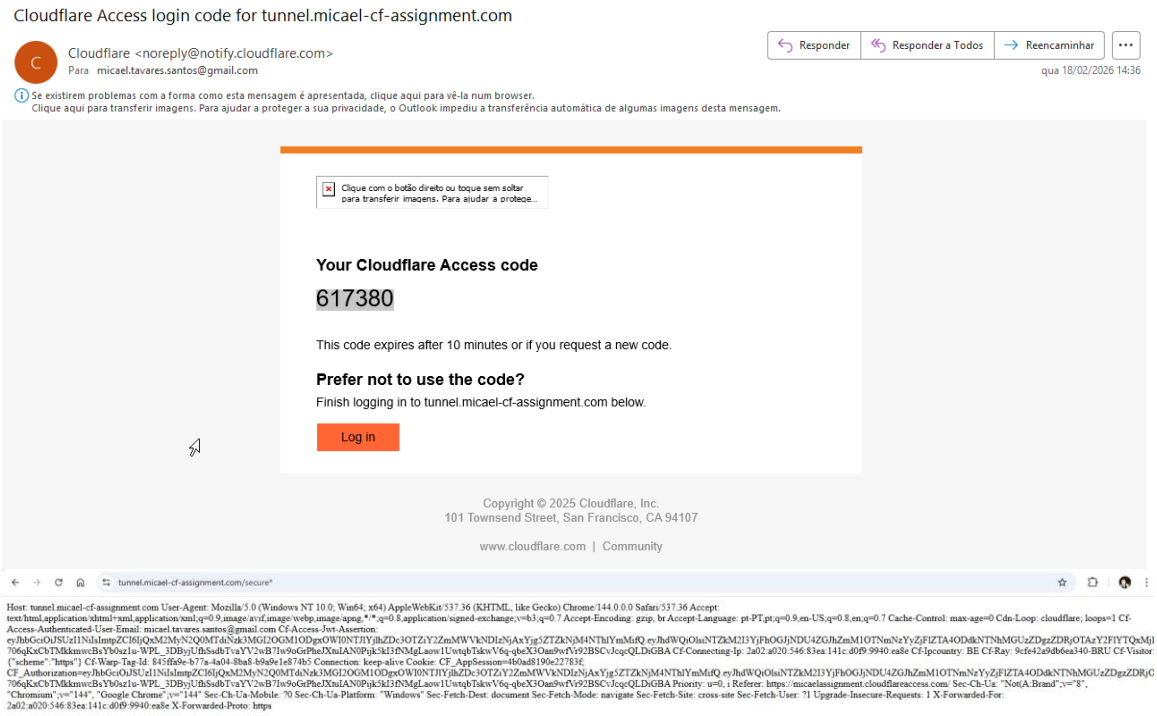


Fig. 20Block test



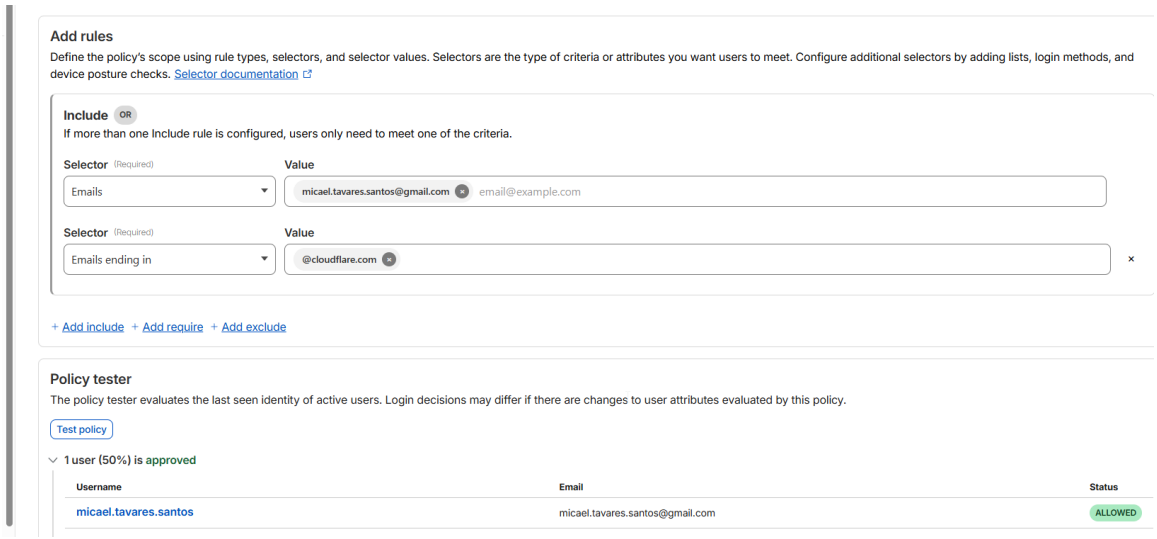


Fig. 21Working Test

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.

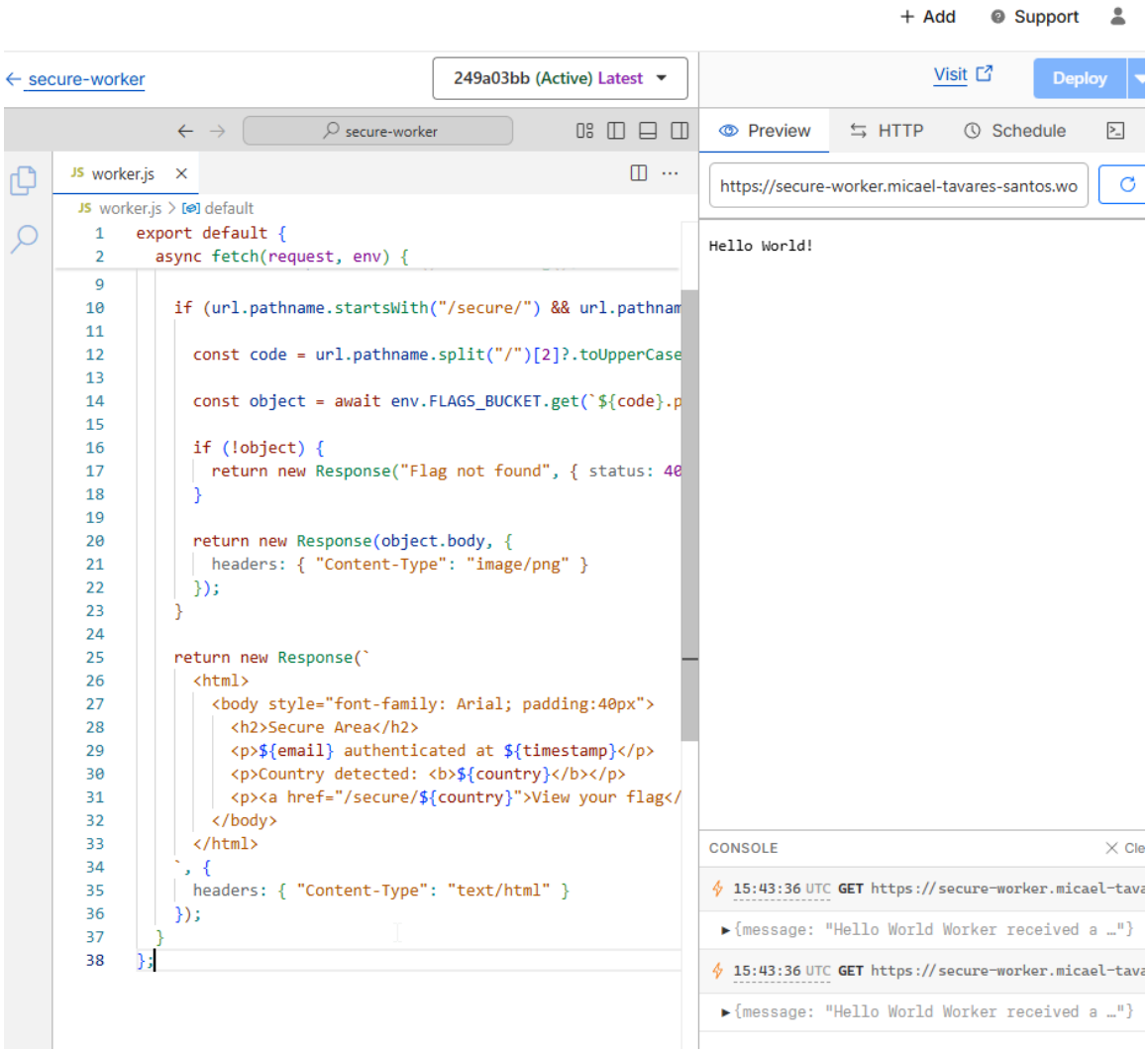


Fig. 22Worker code

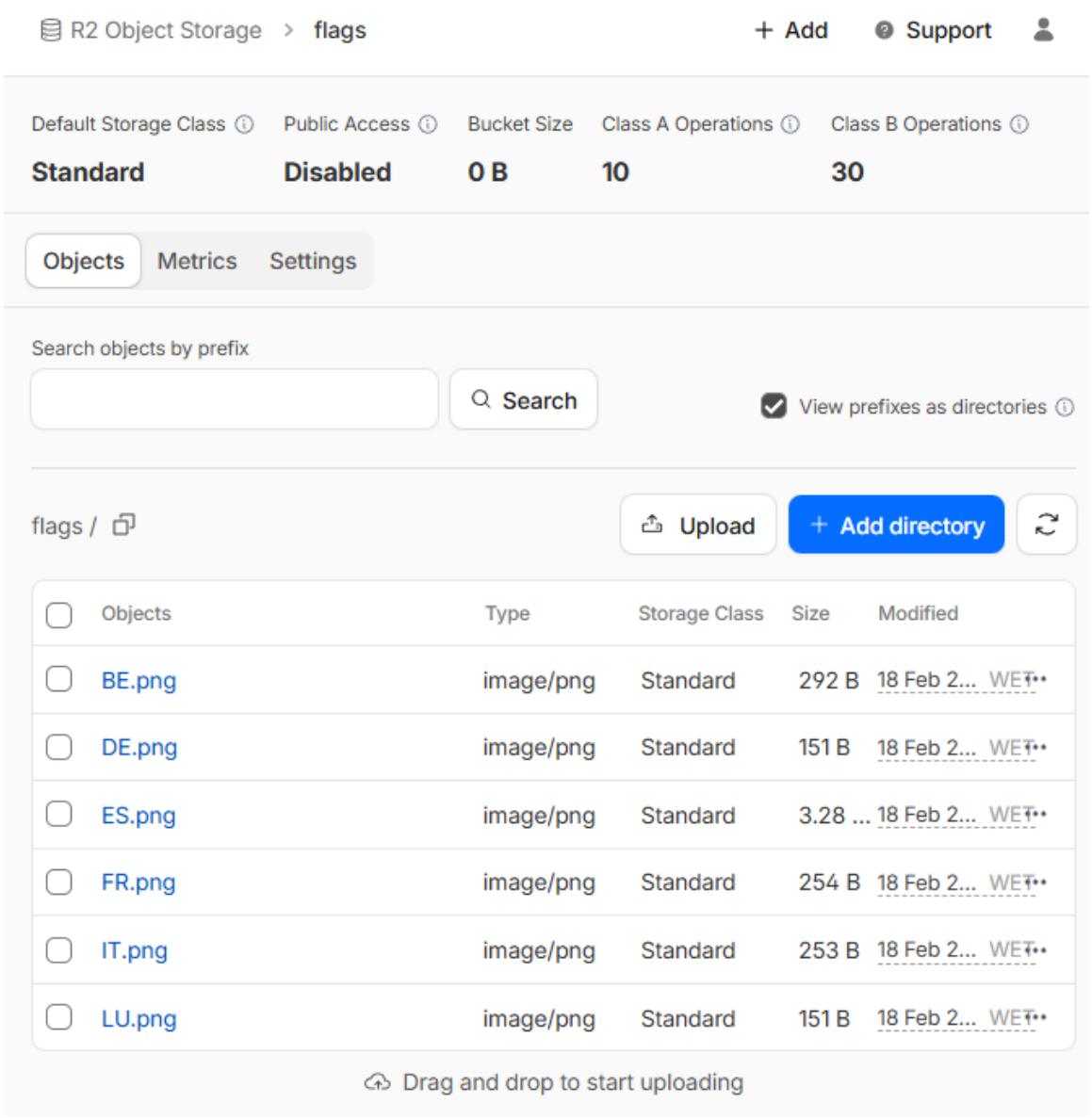


Fig. 23R2 bucket

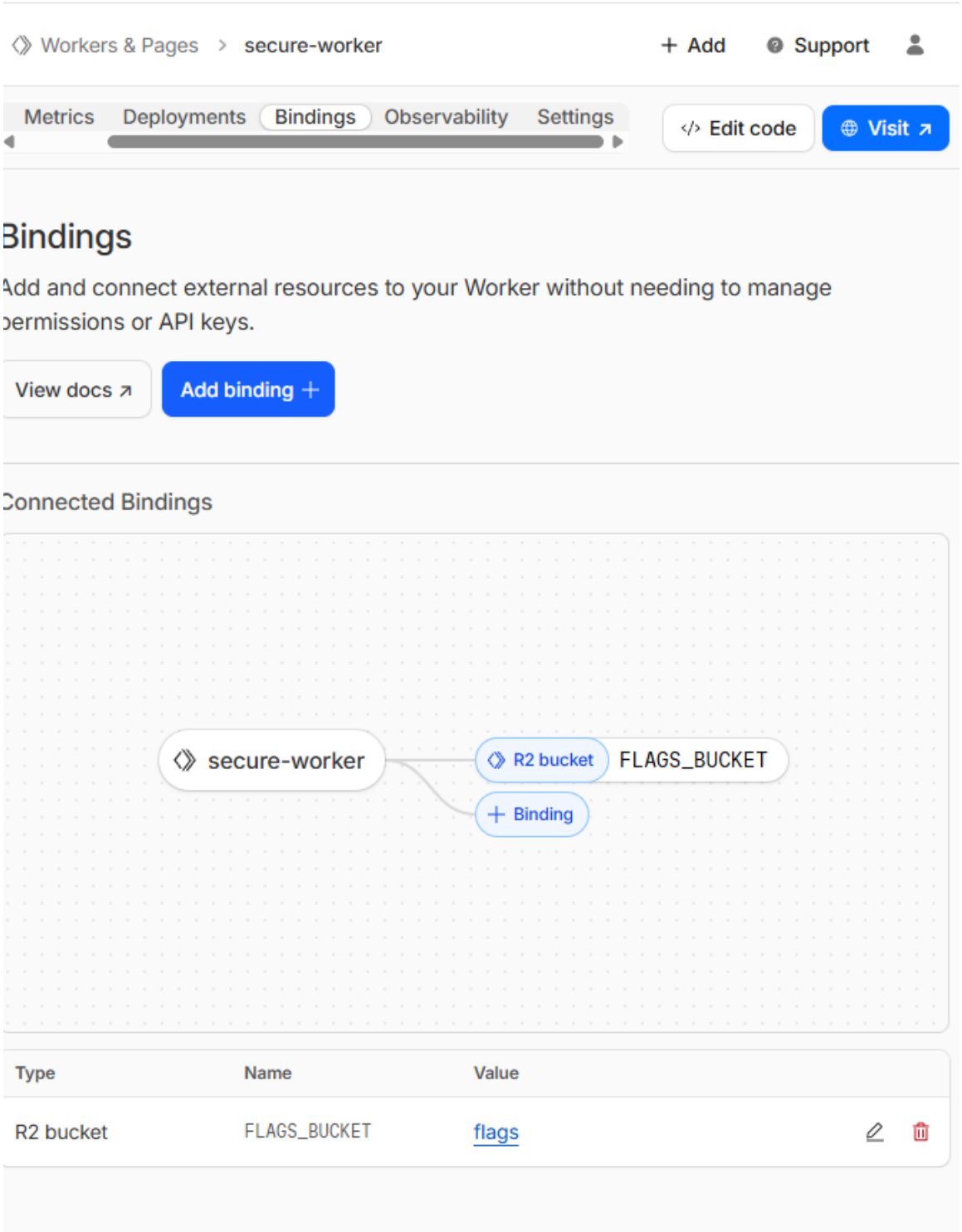


Fig. 24Worker binding

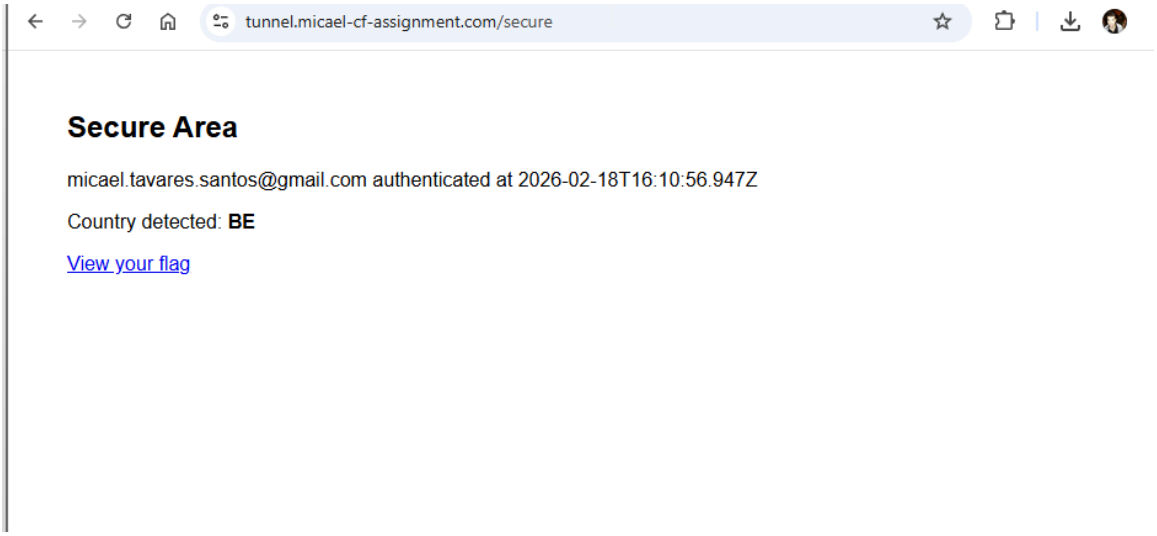


Fig. 25Secure page working

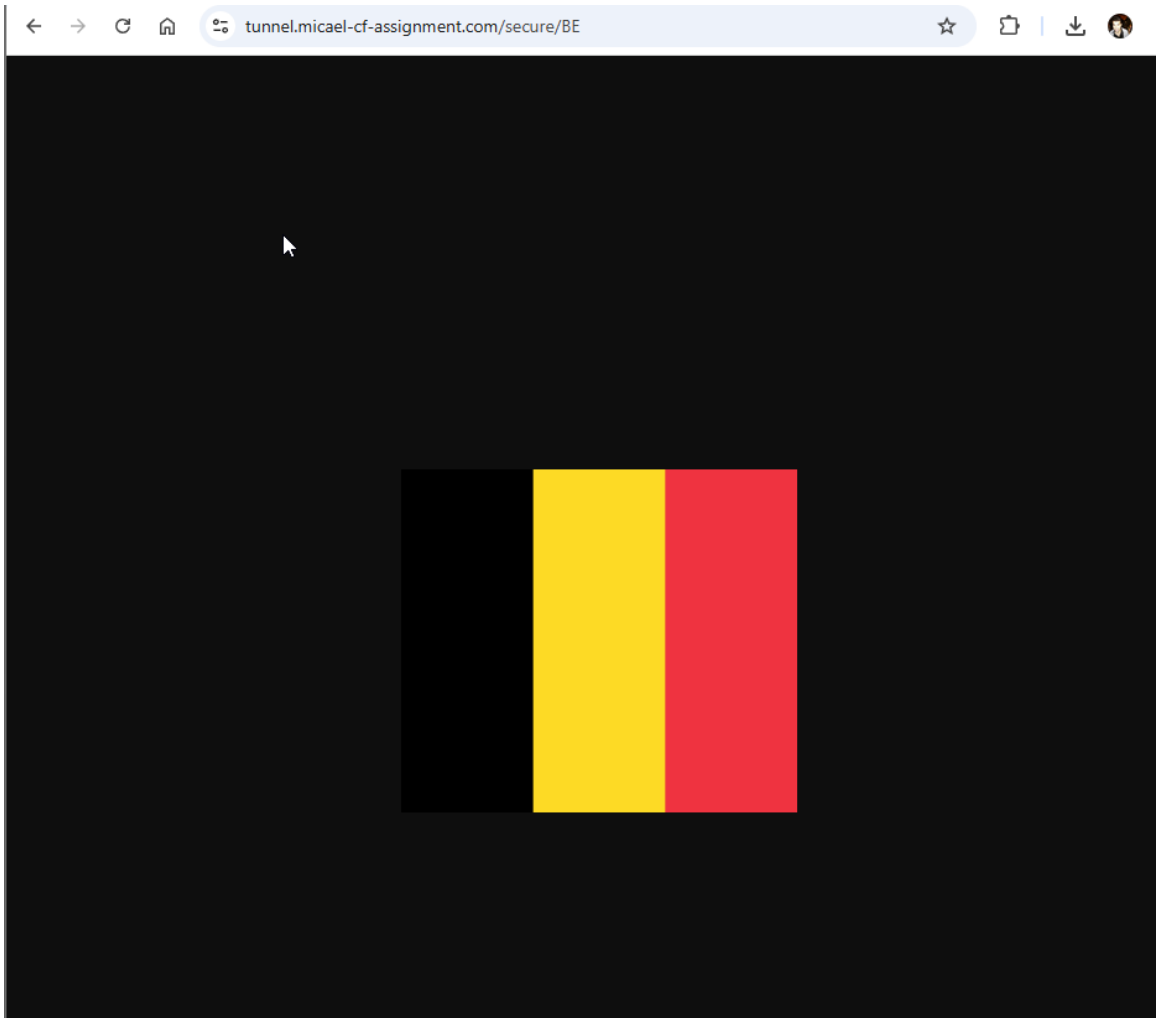


Fig. 26Flag displayed

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