Deploying a Public K3s App via Cloudflare Tunnel (Zero Port Forwarding)

Overview:

This guide documents how to expose services running inside a K3s Kubernetes cluster on Multipass to the public internet securely using Cloudflare Tunnel. No cert-manager, TLS certificates, or router port-forwarding are required. Cloudflare handles all HTTPS termination and zero-trust access via an encrypted tunnel.

Sections:

1. Setup K3s on Multipass

2. Verify Traefik LoadBalancer

3. Deploy Application (Namespace, Deployment, Service)

4. Apply Ingress (HTTP only)

5. Cloudflare DNS Setup

6. Install Cloudflared on K3s Master

7. Create and Configure Tunnel

8. Route Tunnel DNS

9. Configure cloudflared to forward into Traefik

10. Validate internal routing

11. Test the public domain

12. Troubleshooting Cheatsheet

Key Principle:

Cloudflare Tunnel terminates HTTPS outside the cluster. Traffic entering Kubernetes is HTTP only. Therefore, cert-manager, TLS certificates, websecure entrypoints, and ACME HTTP01 challenges are not required.

Clean Ingress Example:

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: explorage-ingress

namespace: explorage

annotations:

kubernetes.io/ingress.class: traefik

spec:

rules:

- host: explorage.pulami.co.uk

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: explorage-backend-svc

port:

number: 8080

Cloudflared Config Example:

/etc/cloudflared/config.yml

tunnel:

credentials-file: /home/ubuntu/.cloudflared/.json

ingress:

- hostname: explorage.pulami.co.uk

service: http://192.168.2.14

- service: http_status:404

Architecture Flow:

Browser (HTTPS)

→ Cloudflare Edge (TLS Termination)

→ Cloudflare Tunnel (Encrypted QUIC)

→ cloudflared on K3s Master (HTTP)

→ Traefik LoadBalancer (192.168.2.14)

→ Ingress Rule

→ Service (ClusterIP)

→ Backend Pod

Troubleshooting:

502 Bad Gateway:

Cause: cloudflared cannot resolve cluster DNS.

Fix: Use Traefik LB IP instead of internal service DNS.

TLS Errors:

Cause: TLS configured incorrectly in cluster.

Fix: Remove TLS; Cloudflare handles HTTPS.

ACME Solver Ingress Blocking Routing:

Fix: Delete ingress resources with acme.cert-manager.io/http01-solver.

Cert-Manager Cleanup:

kubectl delete ingress -n namespace -l acme.cert-manager.io/http01-solver=true

kubectl delete challenges.acme.cert-manager.io --all -n namespace

Cloudflared Restart:

sudo systemctl restart cloudflared

sudo journalctl -u cloudflared -f

Result:

Your app becomes globally accessible with HTTPS, without exposing your home IP or opening router ports. Perfect for homelab or production-grade deployments.