Multistack App on Kubernetes

Deploying a Voting Application to AWS EKS

Python Node.js .NET Redis PostgreSQL

Project Overview

The Challenge

- Multi-language microservices
- Real-time voting system
- Message queue processing
- Production-ready Kubernetes

Tech Stack

Vote: Python/Flask

Worker: .NET 7

Result: Node.js/Express

Queue: Redis

Database: PostgreSQL

Architecture

```
Browser (User)
   ↓
AWS ELB (Auto-provisioned)
   ↓
╔════════════════════════════════════════╗
║   Kubernetes Cluster (EKS)             ║
║                                        ║
║  ┌──────────────────────────────────┐ ║
║  │ NGINX Ingress Controller Pod     │ ║
║  │ (Pre-installed infrastructure)   │ ║
║  └──────────┬──────────────┬────────┘ ║
║             │              │           ║
║  ┌──────────▼──┐  ┌────────▼───┐     ║
║  │ Vote Pod    │  │ Result Pod │     ║
║  │ (Flask)     │  │ (Node.js)  │     ║
║  └──────┬──────┘  └──────┬─────┘     ║
║         │ Writes         │ Reads      ║
║         ↓                ↓            ║
║  ┌───────────┐     ┌────────────┐    ║
║  │ Redis Pod │     │Postgres Pod│    ║
║  └─────┬─────┘     └──────┬─────┘    ║
║        │ Consume          ↑           ║
║        ↓                  │           ║
║  ┌─────────────┐          │           ║
║  │ Worker Pod  │──────────┘           ║
║  │ (.NET)      │  Writes              ║
║  └─────────────┘                      ║
╚════════════════════════════════════════╝
```

CI/CD Pipeline

Trigger: Push to `main` branch

Build Phase:

- Build Docker images (vote, worker, result)
- Push to Docker Hub

Deploy Phase:

- Connect to EKS cluster
- Create Kubernetes secrets
- Apply manifests: `kubectl apply -f K8s/`

Deployment: 7-10 min

Problem 1: Infrastructure Issues

Symptoms:

```
Browser: DNS_PROBE_FINISHED_NXDOMAIN
Worker:  Waiting for db... Giving up
```

Root Causes:

Cluster Migration `ironhack-main` → `ironhack-main-2`

- ELB changed, old DNS invalid

Naming Chaos - Code vs Kubernetes

- Code: `redis`, `db` | K8s: `marty-svc-redis`, `marty-svc-postgres`

Missing Secrets - Database credentials never created

Solution 1: Infrastructure Fixes

Networking

✓ Subdomain routing

✓ vote.marty.ironhack.com

✓ No path rewriting

Security

✓ GitHub Secrets → K8s

✓ Automated injection

Configuration

✓ Environment variables

✓ Service discovery

✓ Proper naming

Ingress

✓ ingressClassName: nginx

✓ Explicit hostnames

Deep Dive: Why CSS Failed to Load

Path-Based Routing Problem

Original Ingress (Broken):

```
- host: marty.ironhack.com
  paths:
    - path: /vote  # App expects to be at root!
```

Request Flow:

```
1. Browser → http://marty.ironhack.com/vote
2. Flask returns HTML with: <link href="/static/style.css">
3. Browser requests → http://marty.ironhack.com/static/style.css
4. Ingress: No route for /static ❌ 404 Error!
```

The App's Perspective:

- Flask thinks it's at path: `/`
- But Ingress places it at: `/vote`
- Static files resolve to: `/static/...` (wrong!)
- Should resolve to: `/vote/static/...`

Solution: Subdomain Routing

Fixed Ingress:

```
- host: vote.marty.ironhack.com
  paths:
    - path: /  # App IS at root now!
```

Request Flow (Working):

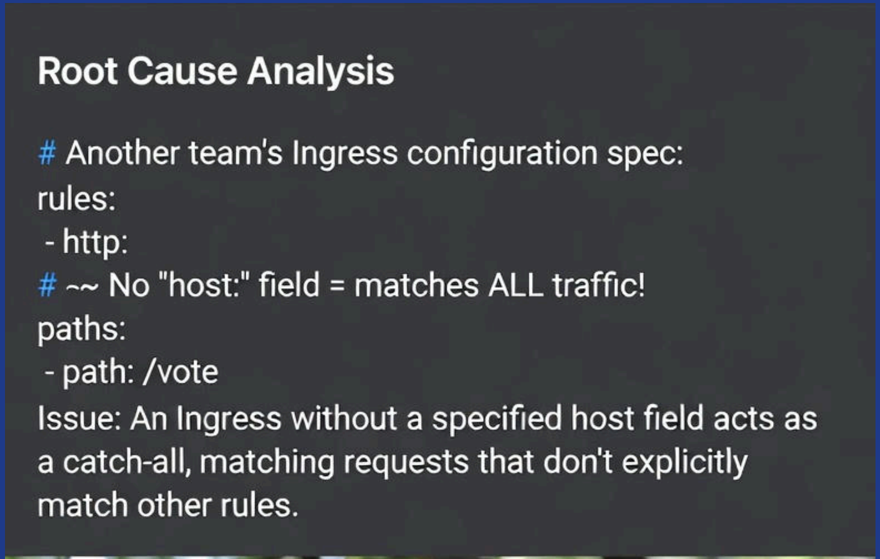
```
1. Browser → http://vote.marty.ironhack.com/
2. Flask returns HTML with: <link href="/static/style.css">
3. Browser requests → http://vote.marty.ironhack.com/static/style.css
4. Ingress routes / → Flask ✅ CSS loads!
```

Why This Works:

- App runs at root path `/`
- Static files at `/static/...` resolve correctly
- No path rewriting needed
- Clean URLs for each service

Problem 2

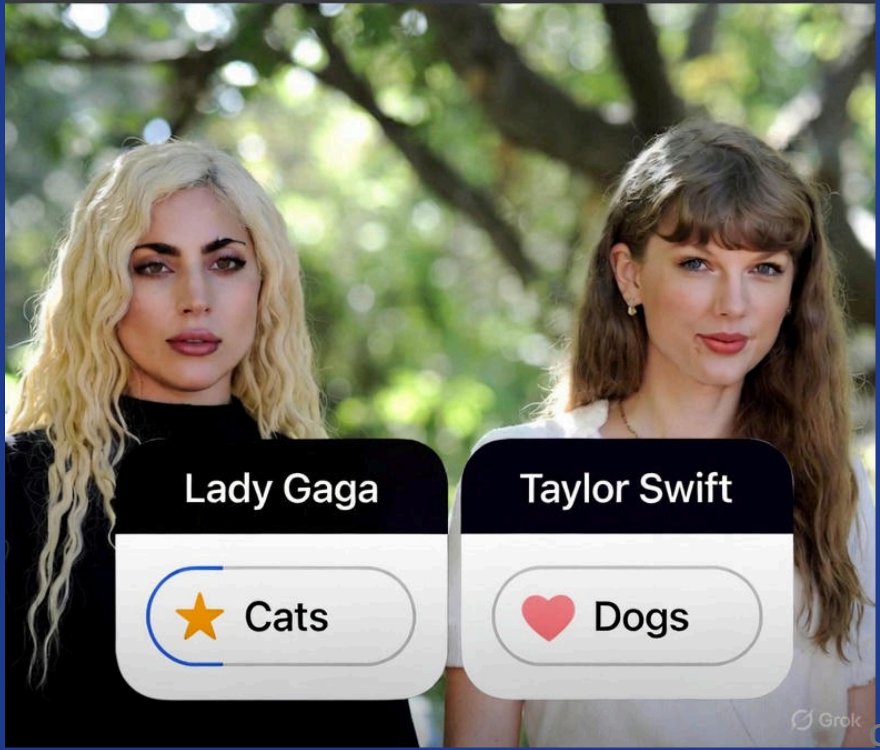
The Wildcard Ingress Mystery



Root Cause Analysis

```
# Another team's Ingress configuration spec:
rules:
 - http:
# ~~ No "host:" field = matches ALL traffic!
paths:
 - path: /vote
```

Issue: An Ingress without a specified host field acts as a catch-all, matching requests that don't explicitly match other rules.



Wildcard Ingress Issue

What Happened?

Accessing `vote.marty.ironhack.com` showed **Taylor Swift vs Lady Gaga** instead of **Cats vs Dogs**.

Root Cause:

```
# Another team's Ingress
spec:
  rules:
  - http:  # No "host:" = catches ALL
      paths:
      - path: /vote
```

Ingress without `host` field acts as catch-all.

Solution 2: Explicit Hosts

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: marty-ingress
spec:
  ingressClassName: nginx
  rules:
  - host: vote.marty.ironhack.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: marty-svc-vote
            port: {number: 80}
```

Key Learning: Always specify explicit `host` values

Problem 3: Hardcoded Connections

Vote (Flask)

```
# Before
Redis(host="redis")

# After
redis_host = os.getenv(
  'REDIS_HOST', 'redis'
)
Redis(host=redis_host)
```

Result (Node.js)

```
// Before
'postgres://user:pass@db'

// After
`postgres://${process.env.POSTGRES_USER}:
${process.env.POSTGRES_PASSWORD}@
${process.env.POSTGRES_HOST}`
```

Problem 3: Worker Variables

Wrong Config

```
env:
  - name: POSTGRES_HOST
  - name: POSTGRES_USER
```

Code expected different names!

Corrected

```
env:
  - name: DB_HOST
    value: "marty-svc-postgres"
  - name: DB_USERNAME
    valueFrom:
      secretKeyRef:
        name: marty-db-credentials
```

Summary

What We Accomplished:

✓ Multi-language microservices on Kubernetes

✓ AWS ELB + NGINX Ingress routing

✓ Secure secret management

✓ Automated CI/CD pipeline

Skills Demonstrated:

Kubernetes - Deployments, Services, Ingress, Secrets
AWS - EKS, ELB
Docker - Multi-stage builds
CI/CD - GitHub Actions

Questions?

Is Kubernetes easy?
Th3BlackHol3
Yes
Yesterday it made me cry

Thank You!

https://github.com/kaiser-data/marty-voting-app

Vote: http://vote.marty.ironhack.com
Result: http://result.marty.ironhack.com