🚀 Performance Monitoring API - Complete Deployment Guide

📋 Table of Contents

🎯 Quick Start

Prerequisites

- GitHub account (for all platforms)
- Git installed locally
- Your enhanced monitoring API code

File Structure

```
performance-monitor-api/
├── app.py                 # Your main API file
├── requirements.txt       # Python dependencies
├── Procfile               # Process file for deployment
├── render.yaml            # Render configuration
├── railway.json           # Railway configuration
├── fly.toml               # Fly.io configuration
├── .env.example           # Example environment variables
└── README.md              # Documentation
```

🌐 Deploy to Render.com (Recommended)

Best for: Free tier with persistent storage, automatic SSL, easy setup

Step 1: Prepare Your Repository

1. Create a new GitHub repository:

```bash
git init
git add .
git commit -m "Initial commit: Performance Monitoring API"
git branch -M main
git remote add origin https://github.com/YOUR_USERNAME/performance-monitor-api.git
git push -u origin main
```

2. Create a `render.yaml` file in your repository root:

```yaml
services:
  - type: web
    name: performance-monitor-api
    env: python
    region: oregon
    plan: free
    buildCommand: pip install -r requirements.txt
    startCommand: gunicorn -w 2 -b 0.0.0.0:$PORT app:app
    envVars:
      - key: PYTHON_VERSION
        value: 3.11.0
      - key: SECRET_KEY
        generateValue: true
      - key: DEBUG
        value: false
      - key: RATE_LIMIT
        value: 100 per hour
    disk:
      name: monitoring-data
      mountPath: /opt/render/project/src/data
      sizeGB: 1
```

Step 2: Deploy on Render

1. Go to render.com and sign up/login

2. Click **"New +" → "Blueprint"**

3. Connect your GitHub repository
4. Render will detect your `render.yaml` and configure everything
5. Click **"Apply"**
6. Wait 5-10 minutes for deployment

Step 3: Get Your API Key

1. Once deployed, go to your service's **Shell** tab
2. Run this command:

```bash
python3 -c "from app import monitor; print(monitor.db.create_api_key('my-first-key'))"
```

3. **SAVE THE API KEY** - you won't see it again!

Step 4: Test Your API

```bash
# Replace YOUR_APP_NAME and YOUR_API_KEY
curl https://YOUR_APP_NAME.onrender.com/api/health

curl -H "X-API-Key: YOUR_API_KEY" \
  https://YOUR_APP_NAME.onrender.com/api/metrics
```

🚂 Deploy to Railway.app

Best for: Fastest deployment, great free tier, excellent DX

Step 1: Setup

1. Push your code to GitHub (same as Render steps above)
2. Create a `railway.json`:

```json
```

```
{
  "$schema": "https://railway.app/railway.schema.json",
  "build": {
    "builder": "NIXPACKS"
  },
  "deploy": {
    "startCommand": "gunicorn -w 2 -b 0.0.0.0:$PORT app:app",
    "restartPolicyType": "ON_FAILURE",
    "restartPolicyMaxRetries": 10
  }
}
```

3. Create a `Procfile`:

```
web: gunicorn -w 2 -b 0.0.0.0:$PORT app:app
```

Step 2: Deploy

1. Go to railway.app
2. Click **"Start a New Project"**
3. Select **"Deploy from GitHub repo"**
4. Choose your repository
5. Railway auto-detects Python and deploys

Step 3: Add Environment Variables

1. Go to your project → **Variables** tab
2. Add these variables:

```
SECRET_KEY=<auto-generated>
DEBUG=false
RATE_LIMIT=100 per hour
DATABASE_PATH=/app/data/monitoring.db
```

Step 4: Add Persistent Volume

1. Go to **Settings → Volumes**
2. Click **"Add Volume"**

3. Mount path: `/app/data`
4. Size: 1GB

Step 5: Generate API Key

1. Go to **Settings → Console**
2. Run:

```bash
python -c "from app import monitor; print(monitor.db.create_api_key('railway-key'))"
```

🛫 Deploy to Fly.io

Best for: Global edge deployment, great performance, free allowance

Step 1: Install Fly CLI

```bash
# macOS
brew install flyctl

# Linux
curl -L https://fly.io/install.sh | sh

# Windows
powershell -Command "iwr https://fly.io/install.ps1 -useb | iex"
```

Step 2: Login and Initialize

```bash
flyctl auth login
flyctl launch
```

Answer the prompts:

- App name: `your-monitor-api` (or auto-generate)
- Region: Choose closest to you
- Database: **No** (we use SQLite)

- Deploy now: **No**

Step 3: Configure fly.toml

Edit the generated `fly.toml`:

toml

```toml
app = "your-monitor-api"
primary_region = "iad"

[build]
  builder = "paketobuildpacks/builder:base"

[env]
  PORT = "8080"
  PYTHON_VERSION = "3.11"

[http_service]
  internal_port = 8080
  force_https = true
  auto_stop_machines = true
  auto_start_machines = true
  min_machines_running = 0

[[vm]]
  cpu_kind = "shared"
  cpus = 1
  memory_mb = 512

[mounts]
  source = "monitoring_data"
  destination = "/data"
```

Step 4: Create Volume and Deploy

bash

```
# Create persistent volume
flyctl volumes create monitoring_data --size 1

# Set environment variables
flyctl secrets set SECRET_KEY=$(openssl rand -hex 32)
flyctl secrets set DEBUG=false

# Deploy
flyctl deploy
```

Step 5: Generate API Key

bash

```
flyctl ssh console
python -c "from app import monitor; print(monitor.db.create_api_key('fly-key'))"
exit
```

Environment Variables

Create a `.env.example` file for reference:

bash

```
# Security
SECRET_KEY=your-secret-key-here

# API Keys (comma-separated for multiple keys)
API_KEYS=

# Database
DATABASE_PATH=monitoring.db

# Performance
MAX_HISTORY_RECORDS=10000

# Rate Limiting
RATE_LIMIT=100 per hour

# Debug (set to false in production)
DEBUG=false
```

For Production:

- Never commit `.env` file to git
- Use platform's secret management
- Generate strong SECRET_KEY: `python -c "import secrets; print(secrets.token_hex(32))"`

🧪 Testing Your Deployment

1. Health Check (No Auth Required)

```bash
curl https://your-api.onrender.com/api/health
```

Expected response:

```json
{
  "status": "healthy",
  "timestamp": "2025-01-27T10:30:00",
  "version": "2.0"
}
```

2. Get Metrics (Auth Required)

```bash
curl -H "X-API-Key: pm_your_api_key_here" \
  https://your-api.onrender.com/api/metrics
```

3. View Errors

```bash
curl -H "X-API-Key: pm_your_api_key_here" \
  "https://your-api.onrender.com/api/errors?limit=10"
```

4. Test Error Logging

```bash
```

```
curl -X POST \
  -H "X-API-Key: pm_your_api_key_here" \
  -H "Content-Type: application/json" \
  -d '{"type":"TEST_ERROR","message":"Testing from client"}' \
  https://your-api.onrender.com/api/test-error
```

5. Performance History

bash

```
curl -H "X-API-Key: pm_your_api_key_here" \
  "https://your-api.onrender.com/api/performance?limit=20"
```

🔒 Security Best Practices

1. API Key Management

python

```
# Generate multiple keys for different clients
# Run in your deployment console:
from app import monitor

# Key for mobile app
mobile_key = monitor.db.create_api_key('mobile_app')
print(f"Mobile: {mobile_key}")

# Key for web app
web_key = monitor.db.create_api_key('web_app')
print(f"Web: {web_key}")

# Key for testing
test_key = monitor.db.create_api_key('testing')
print(f"Test: {test_key}")
```

2. Rate Limiting

The API includes rate limiting by default:

- Most endpoints: 60 requests/minute
- Simulate load: 10 requests/hour

- Overall: 100 requests/hour per IP

3. CORS Configuration

Current config allows all origins. For production, restrict it:

```python
# In app.py, modify CORS setup:
CORS(app, resources={
    r"/api/*": {
        "origins": ["https://yourdomain.com", "https://app.yourdomain.com"],
        "methods": ["GET", "POST"],
        "allow_headers": ["Content-Type", "X-API-Key"]
    }
})
```

4. HTTPS Only

All deployment platforms provide free SSL. Ensure force_https is enabled:

- Render: Automatic
- Railway: Automatic
- Fly.io: Set in `fly.toml`

📊 Monitoring Your Monitor

Check Logs

Render:

```bash
# View live logs in dashboard or:
render logs -t performance-monitor-api
```

Railway:

```bash
# View in dashboard or:
railway logs
```

Fly.io:

```bash
flyctl logs
```

Database Backup

Render:

1. Go to Shell tab
2. Run: `cat monitoring.db > /opt/render/project/src/data/backup.db`
3. Download from persistent disk

Railway:

```bash
railway run python -c "import shutil; shutil.copy('monitoring.db', 'backup.db')"
```

Fly.io:

```bash
flyctl ssh sftp get /data/monitoring.db ./backup.db
```

Success Checklist

- [ ] Code deployed to platform
- [ ] Health check returns `{"status": "healthy"}`
- [ ] API key generated and saved securely
- [ ] Authenticated endpoints working
- [ ] Rate limiting tested
- [ ] Error logging working
- [ ] Persistent storage configured
- [ ] Logs accessible
- [ ] Environment variables set
- [ ] HTTPS enabled

🆘 Troubleshooting

"Module not found" Error

```bash
# Ensure requirements.txt is present and has all dependencies
pip freeze > requirements.txt
git add requirements.txt
git commit -m "Update dependencies"
git push
```

"API Key Invalid" Error

- Regenerate key in console
- Ensure using `X-API-Key` header (not `Authorization`)
- Check for typos in key

"Database locked" Error

- Restart the service
- Check if multiple workers trying to write simultaneously
- Consider reducing worker count in Procfile

High Memory Usage

- Reduce `MAX_HISTORY_RECORDS` in environment variables
- Run cleanup: Access shell and `DELETE FROM metrics WHERE created_at < date('now', '-7 days')`

📞 Next Steps

After deployment:

1. Generate API keys for each client
2. Integrate with your applications (see client library guide)
3. Set up monitoring alerts
4. Create backup schedule
5. Document your API endpoints

Your API is now live and ready to monitor your applications!