DEPLOYMENT.md — Sentinel

Table of Contents

Prerequisites

Required Software

- **Node.js**: v18.x or higher (v20.x recommended)
- **npm**: v9.x or higher
- **SQLite3**: System libraries for better-sqlite3 compilation
- **SSH Access**: Required for VM monitoring functionality
- **Operating System**: Linux (Ubuntu recommended) or macOS

Required Accounts & Access

- SSH keys configured for all VMs being monitored
- Access to TheForge infrastructure network
- Port 3000 available on host machine (Claudinator)

System Dependencies

```
# Ubuntu/Debian
sudo apt-get update

sudo apt-get install -y build-essential python3 sqlite3 libsqlite3-dev


# macOS

brew install sqlite3
```

Environment Variables

The following environment variables should be configured:

Variable	Description	Required	Default	Example
`PORT`	Server listening port	No	3000	`3000`
`NODE_ENV`	Environment mode	No	development	`production`
`DB_PATH`	SQLite database file path	No	`./data/sentinel.db`	`/var/sentinel/sentinel.db`
`LOG_LEVEL`	Application log level	No	info	`debug` , `info` , `warn` , `error`
`SSH_KEY_PATH`	Path to SSH private key for VM access	Yes	-	`/home/user/.ssh/id_rsa`
`ALERT_CHECK_INTERVAL`	Cron schedule for alert checks	No	*/5	*/2
`BACKUP_CHECK_INTERVAL`	Cron schedule for	No	`0 */6`	`0 */4`

	backup checks			
METRICS_RETENTION_DAYS	Days to retain metrics history	No	30	90

Creating .env File

```
cat > .env << EOF
PORT=3000

NODE_ENV=production

DB_PATH=/var/sentinel/sentinel.db

SSH_KEY_PATH=/home/forge/.ssh/id_rsa

LOG_LEVEL=info

ALERT_CHECK_INTERVAL=*/5    *

BACKUP_CHECK_INTERVAL=0 */6

METRICS_RETENTION_DAYS=30

EOF
```

Local Development

Step 1: Clone and Install

```
git clone <repository-url> sentinel
cd sentinel

npm install
```

Step 2: Create Data Directory

```
mkdir -p data
chmod 755 data
```

Step 3: Configure SSH Access

```
# Ensure SSH key has correct permissions
chmod 600 ~/.ssh/id_rsa
```

```
# Test SSH access to a monitored VM

ssh -i ~/.ssh/id_rsa user@monitored-vm.local
```

Step 4: Initialize Database

The database will auto-initialize on first run, but you can verify:

```
sqlite3 data/sentinel.db ".tables"
```

Step 5: Start Development Server

```
# With auto-reload (Node 18+ required)
npm run dev

# Or standard start

npm start
```

The server will start on `http://localhost:3000`

Step 6: Verify Health

```
curl http://localhost:3000/health
# Expected: {"status":"ok","uptime":<seconds
```

Development Workflow

```
# Run with custom port
PORT=3001 npm run dev

# Run with debug logging

LOG_LEVEL=debug npm run dev

# Monitor logs in real-time
```

```
tail -f logs/sentinel.log  # if logging to file
```

Build

This is a Node.js Express application that does not require a build step.

Production Dependencies Only

```
npm install --production
```

Verification

```
# Verify all dependencies are installed
npm list --depth=0

# Check for vulnerabilities

npm audit

# Test the application starts

node src/server.js
```

Output Structure

```
sentinel/
├── src/              # Application source code
├── data/             # SQLite database (runtime generated)
├── node_modules/     # Dependencies
├── package.json      # Project manifest
└── .env             # Environment configuration
```

Deployment Options

Recommended: Bare Metal / VM Deployment (Primary)

Best for: Infrastructure monitoring tools that need host-level access

This application is designed to run directly on **Claudinator** (Ubuntu host) to monitor TheForge infrastructure. Deploy as a systemd service for production reliability.

Setup Process:

```
# 1. Create application user
sudo useradd -r -s /bin/bash -m -d /opt/sentinel sentinel


# 2. Deploy application

sudo mkdir -p /opt/sentinel

sudo cp -r ./* /opt/sentinel/

sudo chown -R sentinel:sentinel /opt/sentinel

# 3. Install dependencies as sentinel user

sudo -u sentinel bash -c 'cd /opt/sentinel && npm install --production'

# 4. Create data directory

sudo mkdir -p /var/sentinel

sudo chown sentinel:sentinel /var/sentinel
```

Systemd Service Configuration:

```
sudo nano /etc/systemd/system/sentinel.service
```

```
[Unit]
Description=Sentinel Infrastructure Monitor

After=network.target

Wants=network-online.target


[Service]

Type=simple

User=sentinel
```

```
Group=sentinel

WorkingDirectory=/opt/sentinel

Environment="NODE_ENV=production"

Environment="PORT=3000"

Environment="DB_PATH=/var/sentinel/sentinel.db"

Environment="SSH_KEY_PATH=/home/sentinel/.ssh/id_rsa"

ExecStart=/usr/bin/node /opt/sentinel/src/server.js

Restart=always

RestartSec=10

StandardOutput=journal

StandardError=journal

SyslogIdentifier=sentinel


[Install]

WantedBy=multi-user.target
```

```
# Enable and start service
sudo systemctl daemon-reload

sudo systemctl enable sentinel

sudo systemctl start sentinel
```

Check status

```
sudo systemctl status sentinel
```

View logs

```
sudo journalctl -u sentinel -f
```

Alternative: Docker Deployment

Best for: Containerized environments, development, or isolated deployments

```
# Build image
docker build -t sentinel:latest .

# Run container
docker run -d \
  --name sentinel \
  -p 3000:3000 \
  -v /var/sentinel:/data \
  -v ~/.ssh:/root/.ssh:ro \
  -e NODE_ENV=production \
  -e DB_PATH=/data/sentinel.db \
  -e SSH_KEY_PATH=/root/.ssh/id_rsa \
  --restart unless-stopped \
  sentinel:latest
```

Alternative: Railway / Render (Not Recommended)

Note: These platforms are not ideal for this application because:

- Requires SSH access to external VMs
- Needs persistent SQLite database storage
- Better suited for traditional VPS deployment

If you must use Railway/Render, use PostgreSQL instead of SQLite and configure SSH keys via secrets.

Alternative: PM2 Process Manager

Best for: Simpler deployment without systemd

```
# Install PM2 globally
npm install -g pm2

# Start application
cd /opt/sentinel
pm2 start src/server.js --name sentinel

# Save PM2 configuration
pm2 save
```

```
# Setup PM2 to start on boot

pm2 startup

# Run the command it outputs


# Monitor

pm2 monit

pm2 logs sentinel
```

Docker

Dockerfile

```
FROM node:20-alpine

# Install build dependencies for better-sqli

RUN apk add --no-cache \
    python3 \
    make \
    g++ \
    sqlite \
    openssh-client

# Create app directory

WORKDIR /app

# Copy package files

COPY package*.json ./

# Install dependencies
```

```
RUN npm ci --production

# Copy application source

COPY src/ ./src/

# Create data directory

RUN mkdir -p /data

# Expose port

EXPOSE 3000

# Set environment variables

ENV NODE_ENV=production \
    PORT=3000 \
    DB_PATH=/data/sentinel.db

# Health check

HEALTHCHECK --interval=30s --timeout=3s --start-period=40s --retries=3 \
  CMD node -e "require('http').get('http://localhost:3000/health', (r) => {process.exit(r.statusCod

# Run as non-root user

USER node

CMD ["node", "src/server.js"]
```

.dockerignore

```
node_modules
npm-debug.log
```

```
.env

.git

.gitignore

data/

*.md

.DS_Store
```

Docker Compose (Optional)

```
version: '3.8'

services:

  sentinel:
    build: .
    container_name: sentinel
    ports:
      - "3000:3000"
    volumes:
      - sentinel-data:/data
      - ~/.ssh:/home/node/.ssh:ro
    environment:
      - NODE_ENV=production
      - DB_PATH=/data/sentinel.db
      - SSH_KEY_PATH=/home/node/.ssh/id_rsa
      - LOG_LEVEL=info
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "node", "-e", "require('http').get('http://localhost:3000/health', (r) => {proc
      interval: 30s
      timeout: 3s
      retries: 3
      start_period: 40s

volumes:

  sentinel-data:
    driver: local
```

```
# Start with docker-compose
docker-compose up -d
```

View logs

```
docker-compose logs -f
```

Stop

```
docker-compose down
```

CI/CD

GitHub Actions Workflow

Create `.github/workflows/deploy.yml`:

```
name: Deploy Sentinel

on:

  push:
    branches: [main, production]
  workflow_dispatch:

jobs:

  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Run security audit
        run: npm audit --audit-level=high

      - name: Check for syntax errors
        run: node --check src/server.js

  deploy:
    needs: test
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/production'

    steps:
      - uses: actions/checkout@v4

      - name: Deploy to Claudinator
        uses: appleboy/ssh-action@master
        with:
          host: ${{ secrets.DEPLOY_HOST }}
          username: ${{ secrets.DEPLOY_USER }}
          key: ${{ secrets.DEPLOY_SSH_KEY }}
          script: |
            cd /opt/sentinel
            git pull origin production
            npm install --production
            sudo systemctl restart sentinel
            sleep 5
            curl -f http://localhost:3000/health || exit 1

  docker:
    needs: test
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'

    steps:
      - uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Login to Docker Hub
```

```
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}

      - name: Build and push
        uses: docker/build-push-action@v5
        with:
          context: .
          push: true
          tags: |
            ${{ secrets.DOCKERHUB_USERNAME }}/sentinel:latest
            ${{ secrets.DOCKERHUB_USERNAME }}/sentinel:${{ github.sha }}
          cache-from: type=registry,ref=${{ secrets.DOCKERHUB_USERNAME }}/sentinel:buildcache
          cache-to: type=registry,ref=${{ secrets.DOCKERHUB_USERNAME }}/sentinel:buildcache,mode=ma
```

Required GitHub Secrets

Configure these in your repository settings (Settings → Secrets → Actions):

- `DEPLOY_HOST`: Claudinator hostname/IP
- `DEPLOY_USER`: SSH user (e.g., `sentinel`)
- `DEPLOY_SSH_KEY`: Private SSH key for deployment
- `DOCKERHUB_USERNAME`: Docker Hub username (if using Docker)
- `DOCKERHUB_TOKEN`: Docker Hub access token (if using Docker)

Manual Deployment Script

Create `scripts/deploy.sh`:

```
#!/bin/bash
set -e


REMOTE_USER="sentinel"


REMOTE_HOST="claudinator.local"


REMOTE_PATH="/opt/sentinel"


echo "🚀 Deploying Sentinel to $REMOTE_HOST..."


# Create deployment package


echo "📦 Creating deployment package..."


tar --exclude='node_modules' \

    --exclude='.git' \
    --exclude='data' \
    --exclude='.env' \
    -czf sentinel-deploy.tar.gz .

# Upload to server
```

```
echo "📤 Uploading to server..."

scp sentinel-deploy.tar.gz $REMOTE_USER@$REMOTE_HOST:/tmp/

# Deploy on server

echo "🔧 Installing on server..."

ssh $REMOTE_USER@$REMOTE_HOST << 'ENDSSH'
  cd /opt/sentinel
  tar -xzf /tmp/sentinel-deploy.tar.gz
  npm install --production
  sudo systemctl restart sentinel
  sleep 5
  curl -f http://localhost:3000/health && echo "✅ Deployment successful!"
ENDSSH

# Cleanup

rm sentinel-deploy.tar.gz

echo "✨ Deployment complete!"
```

```
chmod +x scripts/deploy.sh
./scripts/deploy.sh
```

Monitoring

Application Health Monitoring

Built-in Health Endpoint

```
# Simple health check
curl http://localhost:3000/health

# Detailed status

curl http://localhost:3000/status
```

Uptime Monitoring Services

Configure external monitoring with:

- **UptimeRobot**: https://uptimerobot.com (free tier available)
- **Pingdom**: https://www.pingdom.com
- **StatusCake**: https://www.statuscake.com

Monitor: `http://claudinator:3000/health` every 5 minutes

System Resource Monitoring

With Systemd Journal

```
# View real-time logs
sudo journalctl -u sentinel -f

# View logs from last hour
sudo journalctl -u sentinel --since "1 hour ago"

# View errors only
sudo journalctl -u sentinel -p err

# Export logs
sudo journalctl -u sentinel --since "2024-01-01" > sentinel-logs.txt
```

With PM2

```
# Real-time logs
pm2 logs sentinel

# Monitor CPU/Memory
pm2 monit

# View detailed info
pm2 show sentinel
```

Database Monitoring

```
# Check database size
ls -lh /var/sentinel/sentinel.db


# SQLite statistics


sqlite3 /var/sentinel/sentinel.db "


SELECT


  name,
  (SELECT COUNT(*) FROM sqlite_master WHERE type='table' AND name=t.name) as row_count
FROM sqlite_master t WHERE type='table';


"

# Check for database locks


fuser /var/sentinel/sentinel.db
```

Performance Monitoring

Node.js Built-in Profiling

Add to your application for production profiling:

```
// In src/server.js - add at startup
if (process.env.NODE_ENV === 'production') {


  const v8 = require('v8');
  const fs = require('fs');

  setInterval(() => {
    const stats = {
      memory: process.memoryUsage(),
      cpu: process.cpuUsage(),
      uptime: process.uptime()
    };
    fs.appendFileSync('/var/log/sentinel-stats.log', JSON.stringify(stats) + '\n');
  }, 60000); // Every minute
}
```

Prometheus Metrics (Suggested)

Install `prom-client` for Prometheus-compatible metrics:

```
npm install prom-client
```

Alert Configuration

Create alerting rules for:

- **Service Down**: Health check fails for 2+ minutes
- **High Memory**: Memory usage > 512MB

- **Database Issues**: Database size > 1GB or lock detected
- **SSH Failures**: Cannot connect to monitored VMs

Log Rotation

Configure logrotate for systemd journals:

```
sudo nano /etc/systemd/journald.conf
```

```
[Journal]
SystemMaxUse=500M

SystemMaxFileSize=50M

MaxRetentionSec=2week
```

```
sudo systemctl restart systemd-journald
```

Backup Strategy

```
# Daily database backup script
#!/bin/bash

BACKUP_DIR="/var/backups/sentinel"

mkdir -p $BACKUP_DIR

# Backup with timestamp

sqlite3 /var/sentinel/sentinel.db ".backup $BACKUP_DIR/sentinel-$(date +%Y%m%d-%H%M%S).db"

# Keep only last 7 days

find $BACKUP_DIR -name "sentinel-*.db" -mtime +7 -delete
```

Add to crontab:

```
sudo crontab -e
0 2   * /opt/sentinel/scripts/backup.sh
```

Recommended Monitoring Stack

For Production:

1. **Grafana + Prometheus**: Full metrics dashboard

2. **Loki**: Log aggregation

3. **Alertmanager**: Alert routing and notifications

4. **Node Exporter**: System metrics

Quick Setup:

```
# Install node_exporter for system metrics
wget https://github.com/prometheus/node_exporter/releases/download/v1.7.0/node_exporter-1.7.0.linux

tar xvfz node_exporter-1.7.0.linux-amd64.tar.gz

sudo cp node_exporter-1.7.0.linux-amd64/node_exporter /usr/local/bin/

sudo useradd -rs /bin/false node_exporter


# Create systemd service for node_exporter


sudo nano /etc/systemd/system/node_exporter.service
```

This gives you infrastructure-wide visibility including Sentinel's health.

Related Documentation

- Readme
- Architecture
- Api
- Contributing
