Sentinel

Table of Contents

Real-time infrastructure monitoring and alerting dashboard for distributed systems

!Sentinel Dashboard

Description

Sentinel is a comprehensive infrastructure monitoring platform designed to provide instant visibility across your entire ecosystem. Built for distributed environments, it continuously monitors VMs, Docker containers, services, and backup systems without requiring manual SSH access to individual machines.

The system aggregates health metrics, container status, event logs, and backup verification data into a unified dashboard. With intelligent alerting and self-healing capabilities, Sentinel detects issues proactively and can automatically restart failed containers or trigger recovery procedures. It's designed to run as a centralized monitoring hub that keeps watch over all your infrastructure 24/7.

Whether you're managing a handful of servers or a complex multi-host deployment, Sentinel provides the observability and automation needed to maintain system reliability and reduce operational overhead.

Features

- **Multi-Host Monitoring** — Track status and metrics across all registered hosts from a single dashboard
- **Container Management** — View, restart, and configure auto-restart policies for Docker containers
- **Real-Time Events** — Streaming event log with status tracking and filtering capabilities
- **Alert Management** — Create, configure, and acknowledge infrastructure alerts with notification history
- **Backup Verification** — Monitor backup status and manually trigger backup checks across hosts
- **Self-Healing** — Automatic container restart policies with configurable healing statistics
- **Forge Integration** — Track project deployment tasks, activity, and build progress
- **SSH-Based Monitoring** — Agentless monitoring using SSH connections to remote hosts
- **Health Checks** — Built-in health and status endpoints for uptime monitoring
- **Persistent Storage** — SQLite database for configuration, history, and alert tracking

Quick Start

Prerequisites

- Node.js 18+ installed
- SSH access to monitored hosts
- SQLite3 (bundled with better-sqlite3)

Installation

```
# Clone the repository
git clone <repository-url>


cd sentinel
```

Install dependencies

```
npm install
```

Configure environment

```
cp .env.example .env
```

Edit .env with your SSH credenti

Start the server

```
npm start
```

The dashboard will be available at `http://localhost:3000` (or your configured port).

Usage

Adding a Host

```
curl -X POST http://localhost:3000/api/status/hosts \
  -H "Content-Type: application/json" \
  -d '{
    "hostname": "example-host",
    "ip": "192.168.1.10",
    "sshUser": "admin",
    "sshKey": "/path/to/key"
  }'
```

Checking Container Status

```
// Get all containers for a specific host
fetch('/api/containers/host-123')


  .then(res => res.json())
  .then(containers => console.log(containers));
```

Creating an Alert

```
curl -X POST http://localhost:3000/api/alerts \
  -H "Content-Type: application/json" \
  -d '{
    "type": "container_down",
    "severity": "high",
    "message": "Container nginx-proxy is not running",
    "hostId": "host-123"
  }'
```

Enabling Auto-Restart for a Container

```
curl -X PUT http://localhost:3000/api/containers/host-123/nginx-proxy/auto-r
  -H "Content-Type: application/json" \
  -d '{"enabled": true}'
```

Triggering a Manual Backup Check

```
curl -X POST http://localhost:3000/api/backups/check \
  -H "Content-Type: application/json" \
  -d '{"hostId": "host-123"}'
```

Screenshots Gallery

No additional screenshots provided

Tech Stack

- **Backend Framework:** Express.js 4.21
- **Language:** JavaScript (Node.js)
- **Database:** SQLite3 (via better-sqlite3)
- **SSH Client:** ssh2 for agentless monitoring
- **Task Scheduling:** node-cron for periodic checks
- **Runtime:** Node.js 18+

Key Dependencies

- `express` — Web framework and API routing
- `better-sqlite3` — High-performance SQLite database
- `ssh2` — SSH2 protocol client for remote command execution
- `node-cron` — Cron-based task scheduler for monitoring jobs

Configuration

Sentinel uses environment variables for configuration. Create a `.env` file in the project root:

```
# Server Configuration
PORT=3000

HOST=0.0.0.0
```

Database

```
DB_PATH=./data/sentinel.db
```

SSH Configuration (defaults for

```
SSH_USER=admin

SSH_KEY_PATH=/path/to/ssh/key

SSH_PORT=22
```

Monitoring Intervals

```
CONTAINER_CHECK_INTERVAL=60000  # milliseconds

BACKUP_CHECK_INTERVAL=3600000   # 1 hour
```

```
HEALING_CHECK_INTERVAL=300000   # 5 minutes
```

Alert Configuration

```
ALERT_RETENTION_DAYS=30
```

Database Schema

The SQLite database is automatically initialized on first run. It includes tables for:

- `hosts` — Registered monitoring targets
- `alerts` — Alert definitions and history
- `events` — System event log
- `config` — Application configuration
- `healing_stats` — Self-healing action history

Contributing

Contributions are welcome! Please see CONTRIBUTING.md for guidelines on submitting issues, feature requests, and pull requests.

License

Related Documentation

- Architecture
- Api
- Deployment
- Contributing