Infra Sentinel – Setup & Usage Guide

Version 0.1.0 · Generated on 2025-09-25

1. Project Overview

Infra Sentinel is a professional-grade toolkit skeleton designed for automated incident detection and remediation in SRE and DevOps contexts. It operates on rule-based YAML files and reacts to events like high CPU, CrashLoopBackOff, or rollout failures.

2. Key Features

1. Rule-based engine (YAML) with priorities and cooldowns
2. Dry-run safe (no side effects by default)
3. Audit logs in JSONL format
4. Approval gate mechanism for risky actions
5. Providers for Kubernetes, Slack, HTTP Webhooks (stubbed)
6. Synthetic demo event stream for local testing

3. Folder Structure

- config/: Sentinel config and provider settings. - rules/: Rule definitions matching event patterns. - demo/: Sample events and mock K8s state. - src/: Python package code. - docker/: Container setup. - docs/: Architecture and scenario writeups. - tests/: Minimal test suite.

4. Setup & Run Locally

To run Infra Sentinel in a Python environment: 1. Create and activate a virtual environment: ```bash python -m venv .venv && source .venv/bin/activate ``` 2. Install dependencies in editable mode: ```bash pip install -e .[dev] ``` 3. Run the demo event stream in dry-run mode: ```bash infra-sentinel run --config config/sentinel.yaml --events demo/events.jsonl --dry-run ``` 4. View audit logs in `results/audit.jsonl`

5. Push to GitHub

1. Initialize Git (if not already done): ```bash git init git add . git commit -m "Initial commit: Infra Sentinel" ``` 2. Rename branch and connect remote: ```bash git branch -M main git remote add origin https://github.com/YOUR_USERNAME/infra-sentinel.git git push -u origin main ```

6. Docker Usage

To build and run the container: ```bash docker build -t infra-sentinel:latest . docker run --rm -v "$PWD/results:/app/results" infra-sentinel:latest ```

7. Key CLI Flags

Flag	Description
--config	YAML config file path (e.g., config/sentinel.yaml)
--events	JSONL stream of events (e.g., demo/events.jsonl)

--audit	Optional audit log path (default from config)
--dry-run	Enable safe dry-run mode (default: true)

8. Demo Scenarios

- High CPU or error rate → scale deployment +1 in staging (not prod). - CrashLoopBackOff pod → restart pod and send webhook ticket. - Canary rollout with HTTP 5xx > 2% → request approval, rollback if approved.

9. Audit Logs

All actions and rule matches are logged to JSONL files (e.g., results/audit.jsonl) with timestamps, dry-run status, and action results. These can be parsed for dashboarding or alerts.

10. CI/CD Integration

GitHub Actions runs linting (flake8) and unit tests automatically on every push/PR via `.github/workflows/ci.yml`.