Contributing to Sentinel

Table of Contents

Welcome to Sentinel! We're glad you're interested in contributing to our infrastructure monitoring platform. This project helps keep TheForge ecosystem healthy and visible, and your contributions make it even better.

Development Setup

Prerequisites

- Node.js (v18 or higher recommended)
- SSH access to monitored hosts (for testing monitoring features)
- Basic understanding of Docker and system monitoring concepts

Getting Started

1. **Clone the repository**

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

2. **Install dependencies**

```
npm install
```

3. **Set up configuration**

- Review `src/config/` for configuration files - Ensure you have proper SSH keys for host access - Configure database path in `src/db/` if needed

4. **Start development server**

```
npm run dev
```

This uses Node's `--watch` flag for automatic reloading on file changes.

5. **Verify the setup**

- Server should start on port 3000 - Visit `http://localhost:3000/health` to verify - Check `http://localhost:3000/api/status` for monitoring status

Code Style

JavaScript Conventions

- **ES6+ syntax**: Use modern JavaScript features (arrow functions, async/await, destructuring)
- **Semicolons**: Use semicolons consistently
- **Naming**:

- `camelCase` for variables and functions - `PascalCase` for classes - `UPPER_CASE` for constants

- **Indentation**: 2 spaces (standard for Express projects)
- **Quotes**: Single quotes for strings (except when avoiding escaping)

Express.js Patterns

- Use Express Router for route organization
- Keep route handlers thin; extract business logic to services
- Follow RESTful conventions for API endpoints
- Use middleware for cross-cutting concerns (auth, logging, error handling)

File Organization

```
src/
├── routes/          # Express route handlers

├── services/        # Business logic and external integrations

├── db/             # Database and persistence

├── config/         # Configuration management

└── server.js       # Application entry point
```

Database

- Uses `better-sqlite3` for persistence
- Keep queries in service layer, not routes
- Use prepared statements for parameterized queries
- Consider migrations for schema changes

Making Changes

Branch Naming

- `feature/` - New features (e.g., `feature/add-network-monitoring`)
- `fix/` - Bug fixes (e.g., `fix/container-restart-timeout`)
- `refactor/` - Code improvements without behavior changes
- `docs/` - Documentation updates
- `chore/` - Maintenance tasks (dependencies, configs)

Commit Messages

Follow conventional commit format:

```
<type>: <short summary>

[optional body]


[optional footer]
```

Types:

- `feat:` - New feature
- `fix:` - Bug fix
- `refactor:` - Code refactoring
- `docs:` - Documentation changes
- `chore:` - Maintenance tasks
- `perf:` - Performance improvements

Examples:

```
feat: add CPU usage alerts for containers

fix: resolve SSH connection timeout on slow hosts
```

```
refactor: extract backup checking logic to service


docs: update API endpoint documentation
```

Development Workflow

1. Create a branch from `main`

2. Make your changes

3. Test manually using `npm run dev`

4. Commit with clear messages

5. Push and create a Pull Request

Testing

Manual Testing

Currently, Sentinel relies on manual testing given its infrastructure monitoring nature:

1. **Start the development server**

```
   npm run dev
```

2. **Test core endpoints**

- Health check: `GET /health` - Status monitoring: `GET /api/status` - Container monitoring: `GET /api/containers` - Alerts: `GET /api/alerts`

3. **Test monitoring features**

- Verify SSH connections to hosts work - Check Docker container detection - Validate alert triggering conditions - Test backup status checks

4. **Test edge cases**

- Unavailable hosts - Failed SSH connections - Invalid container states - Database connection issues

What to Test

When contributing, ensure your changes work with:

- **Multiple host configurations** - Test with various VM setups
- **Container states** - Running, stopped, restarting containers

- **Network conditions** - Slow connections, timeouts, SSH failures
- **Error handling** - Graceful degradation when services are unavailable
- **Data persistence** - Database operations work correctly

Future: Automated Testing

We welcome contributions to add automated testing:

- Unit tests for service layer logic
- Integration tests for API endpoints
- Mock SSH connections for testing monitoring
- Database fixture management

Pull Request Process

Before Submitting

- [ ] Code follows the style guidelines above
- [ ] Manual testing completed successfully
- [ ] No console.log() statements left in code (use proper logging)
- [ ] Comments added for complex logic
- [ ] Related documentation updated

PR Template

```
## Description
Brief description of what this PR does

Type of Change

• [ ] Bug fix
• [ ] New feature
• [ ] Refactoring
• [ ] Documentation
• [ ] Chore
```

```
Testing

How was this tested?

Related Issues

Fixes #(issue number)

Screenshots (if applicable)
```

Review Expectations

- PRs require at least one review before merging
- Address review comments promptly
- Keep PRs focused and reasonably sized
- Be responsive to feedback and questions
- CI checks must pass (when implemented)

Merging

- Squash commits for feature branches
- Use meaningful merge commit messages
- Delete branch after merging
- Update related issues and documentation

Issue Reporting

Bug Reports

When filing a bug, include:

1. **Description** - Clear summary of the issue

2. **Environment** - OS, Node version, deployment context

3. **Steps to reproduce -** Detailed reproduction steps
4. **Expected behavior** - What should happen
5. **Actual behavior** - What actually happens
6. **Logs** - Relevant error messages or logs
7. **Screenshots** - If applicable (especially for UI issues)

Template:

```
**Environment:**
- OS: Ubuntu 22.04
- Node: v18.x
- Sentinel version: [commit hash or version]

**Description:**
[Clear description]

**Steps to Reproduce:**
1.
2.
3.

**Expected:**
**Actual:**

**Logs:**
```

Feature Requests

Include:

1. **Use case** - What problem does this solve?
2. **Proposed solution** - How might it work?
3. **Alternatives considered -** Other approaches you've thought about
4. **Additional context** - Any relevant background

Security Issues

Do not file public issues for security vulnerabilities. Contact the maintainers directly with:

- Detailed description of the vulnerability
- Steps to reproduce
- Potential impact assessment
- Suggested fixes (if any)

Code of Conduct

Our Standards

- **Be respectful** - Treat all contributors with respect and kindness
- **Be collaborative** - Work together, help each other learn
- **Be constructive** - Provide helpful feedback, not just criticism
- **Be patient** - Everyone has different experience levels
- **Be inclusive** - Welcome contributors from all backgrounds

Unacceptable Behavior

- Harassment, discrimination, or exclusionary behavior
- Trolling, insulting comments, or personal attacks
- Publishing others' private information
- Other conduct inappropriate for a professional setting

Enforcement

Violations may result in:

- Warning from maintainers
- Temporary ban from the project
- Permanent ban for repeated or serious violations

Report issues to project maintainers privately.

Questions?

- Check existing issues and documentation first
- Review `CLAUDE.md` for project context
- Ask questions in issues or discussions
- Reach out to maintainers for guidance

Thank you for contributing to Sentinel! Your work helps keep TheForge infrastructure monitored, healthy, and visible. 🚀

Related Documentation

- Readme
- Architecture
- Api
- Deployment
