/sentry-claude-settings-audit

Source: `~/.claude/skills/sentry-claude-settings-audit/SKILL.md`

---

name: claude-settings-audit description: Analyze a repository to generate recommended Claude Code settings.json permissions. Use when setting up a new project, auditing existing settings, or determining which read-only bash commands to allow. Detects tech stack, build tools, and monorepo structure.

Claude Settings Audit

Analyze this repository and generate recommended Claude Code `settings.json` permissions for read-only commands.

Phase 1: Detect Tech Stack

Run these commands to detect the repository structure:

```
ls -la
find . -maxdepth 2 \( -name "*.toml" -o -name "*.json" -o -name "*.lock" -o -name "*.yaml" -o -name "*.yml" -o -name "Makefile" -o -name "Dockerfile" -o -name "*.tf" \) 2>/dev/null | head -50
```

Check for these indicator files:

Category	Files to Check
Python	`pyproject.toml`, `setup.py`, `requirements.txt`, `Pipfile`, `poetry.lock`, `uv.lock`
Node.js	`package.json`, `package-lock.json`, `yarn.lock`, `pnpm-lock.yaml`
Go	`go.mod`, `go.sum`
Rust	`Cargo.toml`, `Cargo.lock`
Ruby	`Gemfile`, `Gemfile.lock`
Java	`pom.xml`, `build.gradle`, `build.gradle.kts`
Build	`Makefile`, `Dockerfile`, `docker-compose.yml`
Infra	`*.tf` files, `kubernetes/`, `helm/`
Monorepo	`lerna.json`, `nx.json`, `turbo.json`, `pnpm-workspace.yaml`

Phase 2: Detect Services

Check for service integrations:

Service	Detection
Sentry	`sentry-sdk` in deps, `@sentry/*` packages, `.sentryclirc`, `sentry.properties`
Linear	Linear config files, `.linear/` directory

Read dependency files to identify frameworks:

- `package.json` → check `dependencies` and `devDependencies`
- `pyproject.toml` → check `[project.dependencies]` or `[tool.poetry.dependencies]`
- `Gemfile` → check gem names
- `Cargo.toml` → check `[dependencies]`

Phase 3: Check Existing Settings

```
cat .claude/settings.json 2>/dev/null || echo "No existing settings"
```

Phase 4: Generate Recommendations

Build the allow list by combining:

Baseline Commands (Always Include)

```
[
  "Bash(ls:*)",
  "Bash(pwd:*)",
  "Bash(find:*)",
  "Bash(file:*)",
  "Bash(stat:*)",
  "Bash(wc:*)",
  "Bash(head:*)",
  "Bash(tail:*)",
  "Bash(cat:*)",
  "Bash(tree:*)",
  "Bash(git status:*)",
  "Bash(git log:*)",
  "Bash(git diff:*)",
  "Bash(git show:*)",
  "Bash(git branch:*)",
  "Bash(git remote:*)",
  "Bash(git tag:*)",
  "Bash(git stash list:*)",
  "Bash(git rev-parse:*)",
  "Bash(gh pr view:*)",
  "Bash(gh pr list:*)",
  "Bash(gh pr checks:*)",
  "Bash(gh pr diff:*)",
  "Bash(gh issue view:*)",
  "Bash(gh issue list:*)",
  "Bash(gh run view:*)",
```

```
  "Bash(gh run list:*)",
  "Bash(gh run logs:*)",
  "Bash(gh repo view:*)",
  "Bash(gh api:*)"
]
```

Stack-Specific Commands

Only include commands for tools actually detected in the project.

Python (if any Python files or config detected)

If Detected	Add These Commands
Any Python	`python --version`, `python3 --version`
`poetry.lock`	`poetry show`, `poetry env info`
`uv.lock`	`uv pip list`, `uv tree`
`Pipfile.lock`	`pipenv graph`
`requirements.txt` (no other lock)	`pip list`, `pip show`, `pip freeze`

Node.js (if package.json detected)

If Detected	Add These Commands
Any Node.js	`node --version`
`pnpm-lock.yaml`	`pnpm list`, `pnpm why`
`yarn.lock`	`yarn list`, `yarn info`, `yarn why`
`package-lock.json`	`npm list`, `npm view`, `npm outdated`
TypeScript (`tsconfig.json`)	`tsc --version`

Other Languages

If Detected	Add These Commands
`go.mod`	`go version`, `go list`, `go mod graph`, `go env`
`Cargo.toml`	`rustc --version`, `cargo --version`, `cargo tree`, `cargo metadata`
`Gemfile`	`ruby --version`, `bundle list`, `bundle show`
`pom.xml`	`java --version`, `mvn --version`, `mvn dependency:tree`
`build.gradle`	`java --version`, `gradle --version`, `gradle dependencies`

Build Tools

If Detected	Add These Commands
`Dockerfile`	`docker --version`, `docker ps`, `docker images`
`docker-compose.yml`	`docker-compose ps`, `docker-compose config`
`*.tf` files	`terraform --version`, `terraform providers`, `terraform state list`
`Makefile`	`make --version`, `make -n`

Skills (for Sentry Projects)

If this is a Sentry project (or sentry-skills plugin is installed), include:

```
[
  "Skill(sentry-skills:commit)",
  "Skill(sentry-skills:create-pr)",
  "Skill(sentry-skills:code-review)",
  "Skill(sentry-skills:find-bugs)",
  "Skill(sentry-skills:iterate-pr)",
  "Skill(sentry-skills:claude-settings-audit)",
  "Skill(sentry-skills:agents-md)",
  "Skill(sentry-skills:brand-guidelines)",
  "Skill(sentry-skills:doc-coauthoring)",
  "Skill(sentry-skills:security-review)",
  "Skill(sentry-skills:django-perf-review)",
  "Skill(sentry-skills:code-simplifier)",
  "Skill(sentry-skills:skill-creator)",
  "Skill(sentry-skills:skill-scanner)"
]
```

WebFetch Domains

Always Include (Sentry Projects)

```
[
  "WebFetch(domain:docs.sentry.io)",
  "WebFetch(domain:develop.sentry.dev)",
  "WebFetch(domain:docs.github.com)",
  "WebFetch(domain:cli.github.com)"
```

```
]
```

Framework-Specific

If Detected	Add Domains
Django	`docs.djangoproject.com`
Flask	`flask.palletsprojects.com`
FastAPI	`fastapi.tiangolo.com`
React	`react.dev`
Next.js	`nextjs.org`
Vue	`vuejs.org`
Express	`expressjs.com`
Rails	`guides.rubyonrails.org`, `api.rubyonrails.org`
Go	`pkg.go.dev`
Rust	`docs.rs`, `doc.rust-lang.org`
Docker	`docs.docker.com`
Kubernetes	`kubernetes.io`
Terraform	`registry.terraform.io`

MCP Server Suggestions

MCP servers are configured in `.mcp.json` (not `settings.json`). Check for existing config:

```
cat .mcp.json 2>/dev/null || echo "No existing .mcp.json"
```

Sentry MCP (if Sentry SDK detected)

Add to `.mcp.json` (replace `{org-slug}` and `{project-slug}` with your Sentry organization and project slugs):

```
{
  "mcpServers": {
    "sentry": {
      "type": "http",
      "url": "https://mcp.sentry.dev/mcp/{org-slug}/{project-slug}"
    }
  }
}
```

Linear MCP (if Linear usage detected)

Add to `.mcp.json`:

```
{
  "mcpServers": {
    "linear": {
      "command": "npx",
      "args": ["-y", "@linear/mcp-server"],
      "env": {
        "LINEAR_API_KEY": "${LINEAR_API_KEY}"
      }
    }
  }
}
```

Note: Never suggest GitHub MCP. Always use `gh` CLI commands for GitHub.

Output Format

Present your findings as:

1. **Summary Table** - What was detected
2. **Recommended settings.json** - Complete JSON ready to copy
3. **MCP Suggestions** - If applicable
4. **Merge Instructions** - If existing settings found

Example output structure:

```
## Detected Tech Stack

| Category        | Found          |
| --------------- | -------------- |
| Languages       | Python 3.x     |
| Package Manager | poetry         |
| Frameworks      | Django, Celery |
| Services        | Sentry         |
| Build Tools     | Docker, Make   |

## Recommended .claude/settings.json
```

```
\`\`\`json
{
"permissions": {
"allow": [
// ... grouped by category with comments
],
"deny": []
}
}
\`\`\`

## Recommended .mcp.json (if applicable)

If you use Sentry or Linear, add the MCP config to `.mcp.json`...
```

Important Rules

What to Include

- Only READ-ONLY commands that cannot modify state
- Only tools that are actually used by the project (detected via lock files)
- Standard system commands (ls, cat, find, etc.)
- The `:*` suffix allows any arguments to the base command

What to NEVER Include

- **Absolute paths** - Never include user-specific paths like `/home/user/scripts/foo` or `/Users/name/bin/bar`
- **Custom scripts** - Never include project scripts that may have side effects (e.g., `./scripts/deploy.sh`)
- **Alternative package managers** - If the project uses pnpm, do NOT include npm/yarn commands
- **Commands that modify state** - No install, build, run, write, or delete commands

Package Manager Rules

Only include the package manager actually used by the project:

If Detected	Include	Do NOT Include
`pnpm-lock.yaml`	pnpm commands	npm, yarn
`yarn.lock`	yarn commands	npm, pnpm
`package-lock.json`	npm commands	yarn, pnpm
`poetry.lock`	poetry commands	pip (unless also has requirements.txt)
`uv.lock`	uv commands	pip, poetry
`Pipfile.lock`	pipenv commands	pip, poetry

If multiple lock files exist, include only the commands for each detected manager.

Revision #5
Created 2026-02-18 08:39:58 UTC by John
Updated 2026-06-21 20:01:05 UTC by John