OpenSpec

https://dev.to/webdeveloperhyper/how-to-make-ai-follow-your-instructions-more-for-free-openspec-2c85

install

```
yay -S openspec
```

| ~/.kilocode/workflows/openspec-refine.md

```
<!-- OPENSPEC:START -->
### Workflow: OpenSpec Refine Change Proposal

This workflow is designed to update the detailed specifications (spec deltas) and the implementation task list (tasks.md) within an existing OpenSpec change folder after the main proposal.md has been modified.

**Input:**
- A valid OpenSpec change ID (e.g., 'add-user-profiles').

**Steps:**
1. **Analyze Proposal:** Read the contents of the `openspec/changes/<change-id>/proposal.md` file to understand the new or modified requirements.
2. **Update Spec Delta:** Using the new requirements from the proposal, carefully modify the spec delta files (located in `openspec/changes/<change-id>/specs/`) to include, modify, or remove requirements and scenarios.
3. **Update Tasks:** Review and update the `openspec/changes/<change-id>/tasks.md` file, ensuring the checklist of implementation steps is accurate, complete, and aligned with the newly refined spec delta.
4. **Validation (Optional):** After updating, run the `openspec validate <change-id>` command in the terminal to check for any structural errors in the spec deltas. Inform the user of the validation result.
5. **Confirmation:** Confirm to the user that the specs and tasks have been successfully refined based on the proposal updates.

**Goal:** Synchronize the spec deltas and implementation tasks with the latest version of the change proposal.
<!-- OPENSPEC:END -->
```

add openspec to existing project

cd <project>

```
openspec init
```

from agent auto compile **openspec\project.md** with command

```
Please read openspec/project.md and help me fill out with details about my
project, tech stack, and conventions.
```

it is possible to change openspec tools with “openspec init” and “openspec update”

add new feature

Proposal

1) create proposal

```
# kilocode
/openspec-proposal.md add data lake folder common for python and nextjs
backend

# claude and other agents
/opsx:propose add data lake folder common for python and nextjs backend
```

file created

```
openspec/changes/
-- add-data-lake-folder
   |-- design.md
   |-- proposal.md
   |-- specs
   |   `-- data-lake
   |       `-- spec.md
   `-- tasks.md
```

Proposal Structure:

- **proposal.md**: Explains why multiple models are needed (flexibility for different scenarios), what changes will be made, and the impact
- **tasks.md**: Detailed implementation checklist broken into 5 phases (design, configuration, processor modification, main function updates, and testing)
- **spec.md**: Specification deltas with both MODIFIED (existing SVM loading requirement) and ADDED (new model configuration requirement) sections, including proper scenarios

2) **Review proposal.md** and rebuild spec.md and tasks.md

```
update add-data-lake-folder from proposal
```

3) **Review spec.md and tasks.md**

eventually check status of proposals

```
openspec view
openspec list    # Confirm the change folder exists
openspec validate load-multiple-svm-models
openspec show load-multiple-svm-models   # Review proposal, tasks, and spec
delta
```

Apply

coding

```
# kilocode
/openspec-apply.md load-multiple-svm-models

# other agents
/opsx:apply
```

...testing...

Update tasks list

```
All tasks for the change [your-change-name] are finished. Please update the
tasks.md
```

Archive

After implementation and testing, archive the change. All of the checkboxes in the TODO list were checked, and files were moved to the archive folder.

```
#openspec archive load-multiple-svm-models --yes
openspec archive load-multiple-svm-models --skip-specs --yes
```