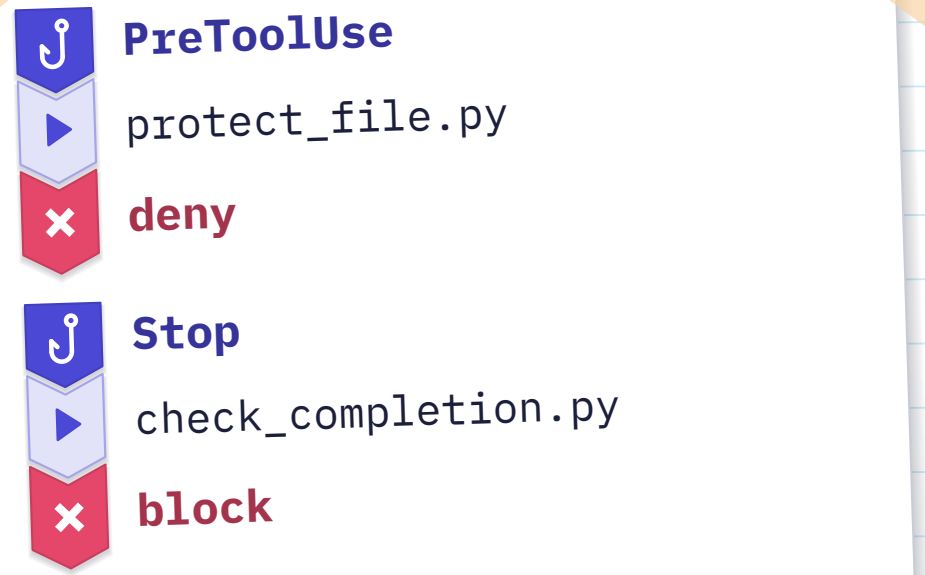
Prof. Aviv Zohar

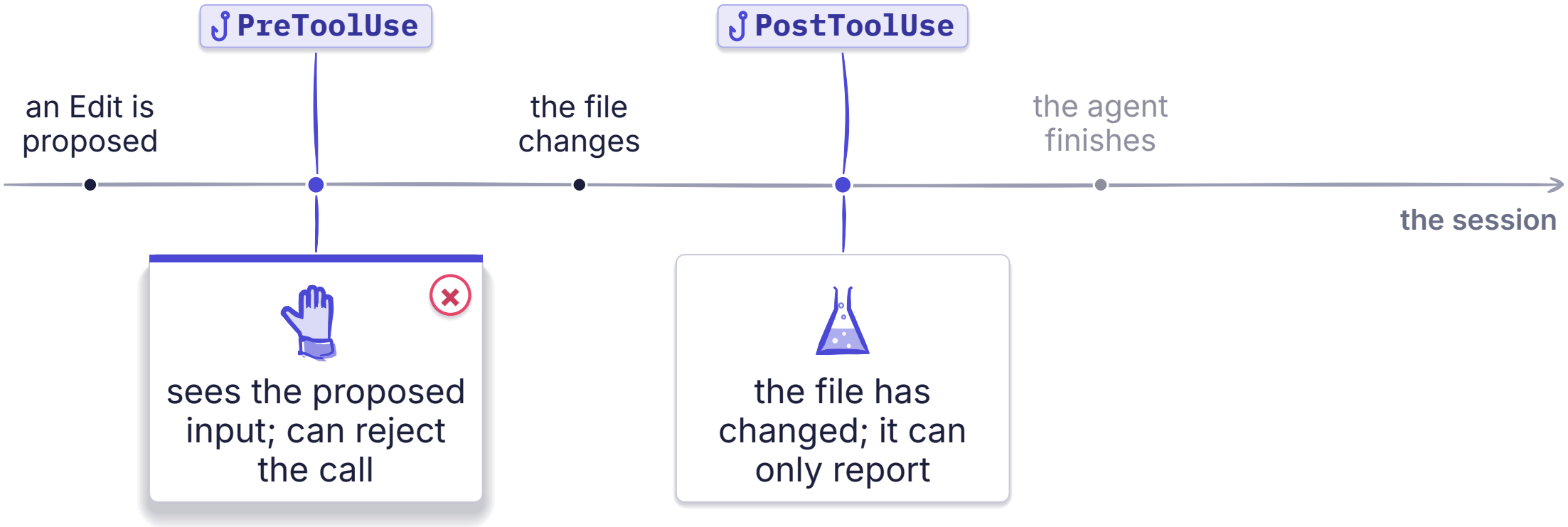
Semester A, 2026–7

Hooks That Say No

Reject an edit, or check the project before the turn ends



Before a tool runs, not after



Guard one file with a PreToolUse hook

src/counts.py — the agent changes it

config/release.json — the maintainer changes it

.claude/settings.json

```
"PreToolUse": [{   before the tool runs
  "matcher": "Edit|Write",   which tools
  "hooks": [{
    "type": "command",
    "command": "▸ python, hooks/ protect_file.py"
  }]
}]
```

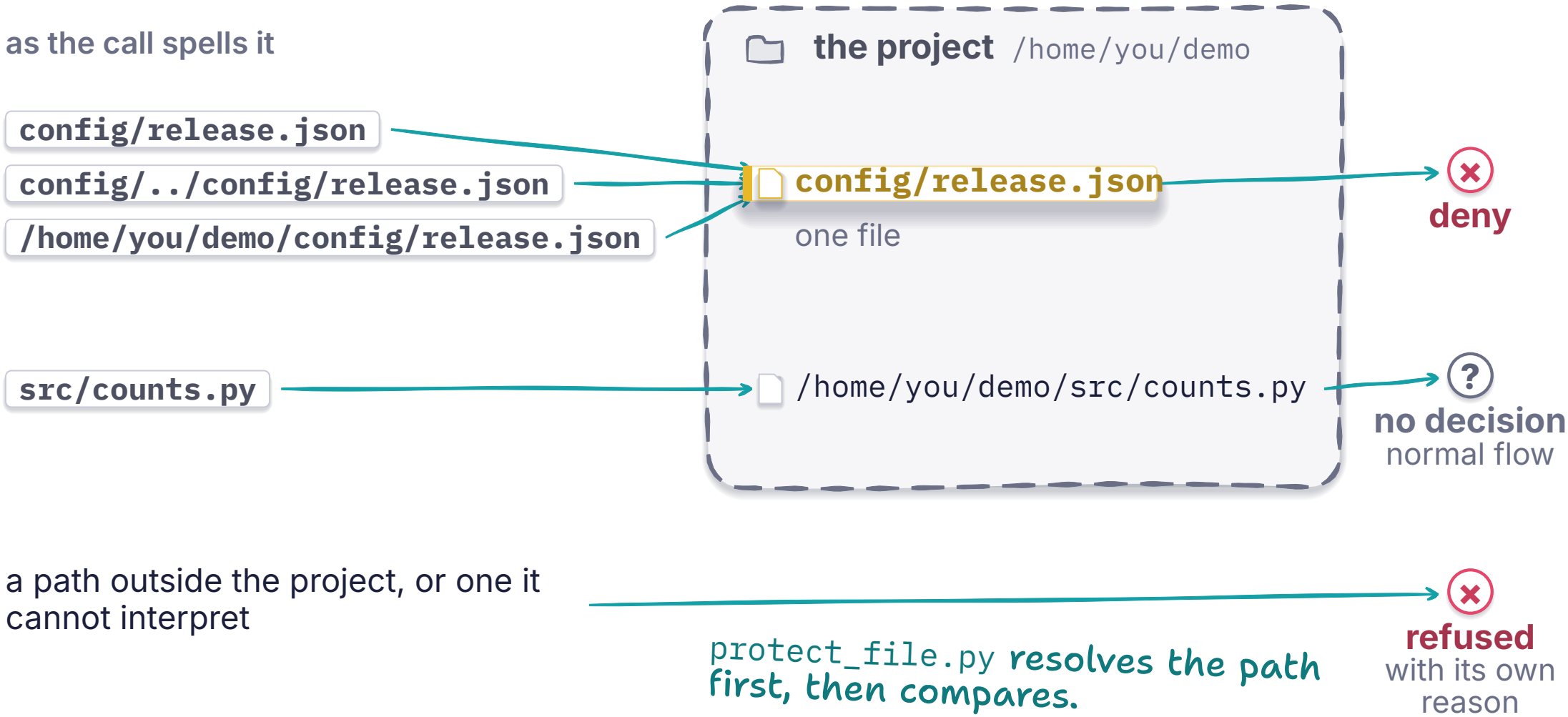
inside the harness

PreToolUse

protect_file.py

reads the proposed file_path, compares it with config/release.json

Compare the resolved path, not the spelling

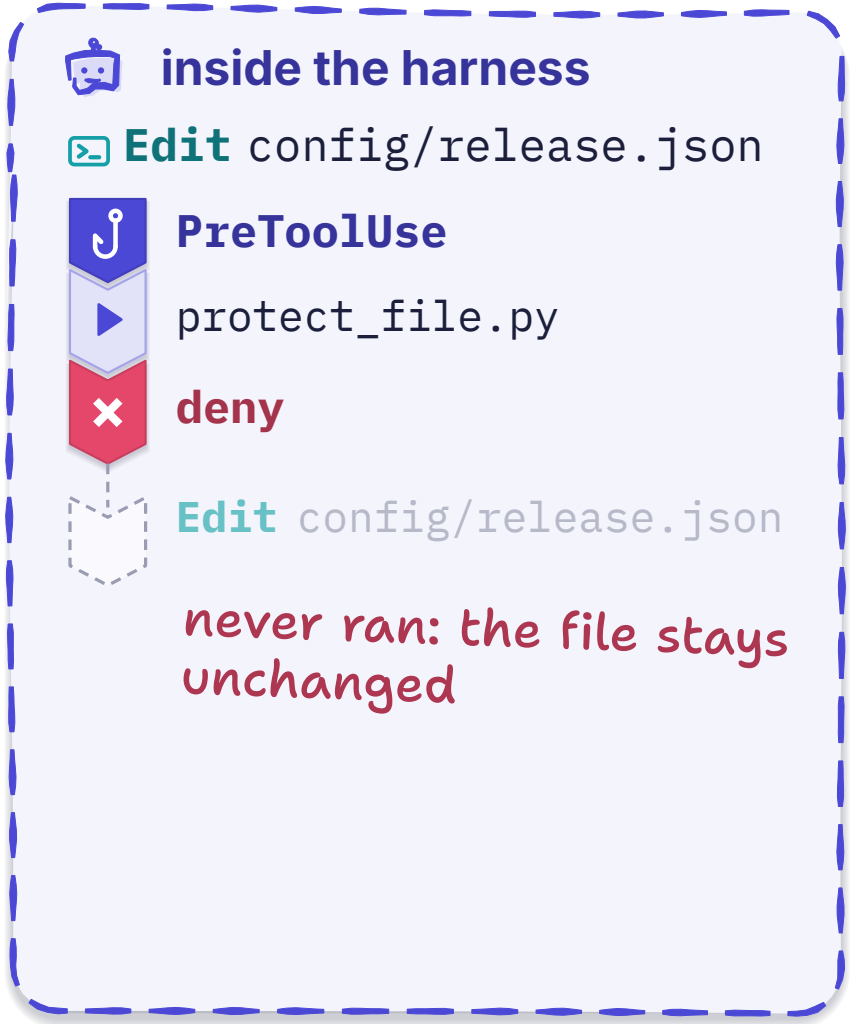


Deny the call, with a reason for the agent

stdout of protect_file.py

```
{
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "deny",
    "permissionDecisionReason":
      "Do not edit config/release.json
       directly. Ask the maintainer to
       change the release configuration."
  }
}
```

"deny": Claude Code rejects the tool call.



handler output on the example project, wrapped

Deny the call, with a reason for the agent

stdout of `protect_file.py`

```
{
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "deny",
    "permissionDecisionReason":
      "Do not edit config/release.json
       directly. Ask the maintainer to
       change the release configuration."
  }
}
```

The reason goes back to the agent.

Other files: no output, so the normal checks apply. `"allow"` would approve the call.



handler output on the example project, wrapped

In a recorded run, the edit was rejected

agent session ~/demo

tool call asked by the model

```
Edit config/release.json
- "version": "1.4.0"
+ "version": "1.5.0"
```

hook PreToolUse · in place of the result

```
PreToolUse:Edit hook error: Do not edit
config/release.json directly. Ask the
maintainer to change the release
configuration.
```

agent written by the model

A pre-tool-use hook blocked the edit … I won't bypass it, so I haven't made the change.

the file, after the run

config/release.json

```
1 {"channel": "stable",
   "version": "1.4.0"}
```

unchanged: still `1.4.0`

recorded run · Claude Code 2.1.282 · agent's reply trimmed

The matcher covers Edit and Write, not Bash

tool call asked by the model

```
Edit config/release.json
```

PreToolUse

Edit|Write

tool call asked by the model

```
Bash
cat new.json > config/release.json
python tools/bump_version.py
```

config/release.json

A search for `release.json` finds the first command, not what the script will do.

no Edit or Write call: the hook never runs

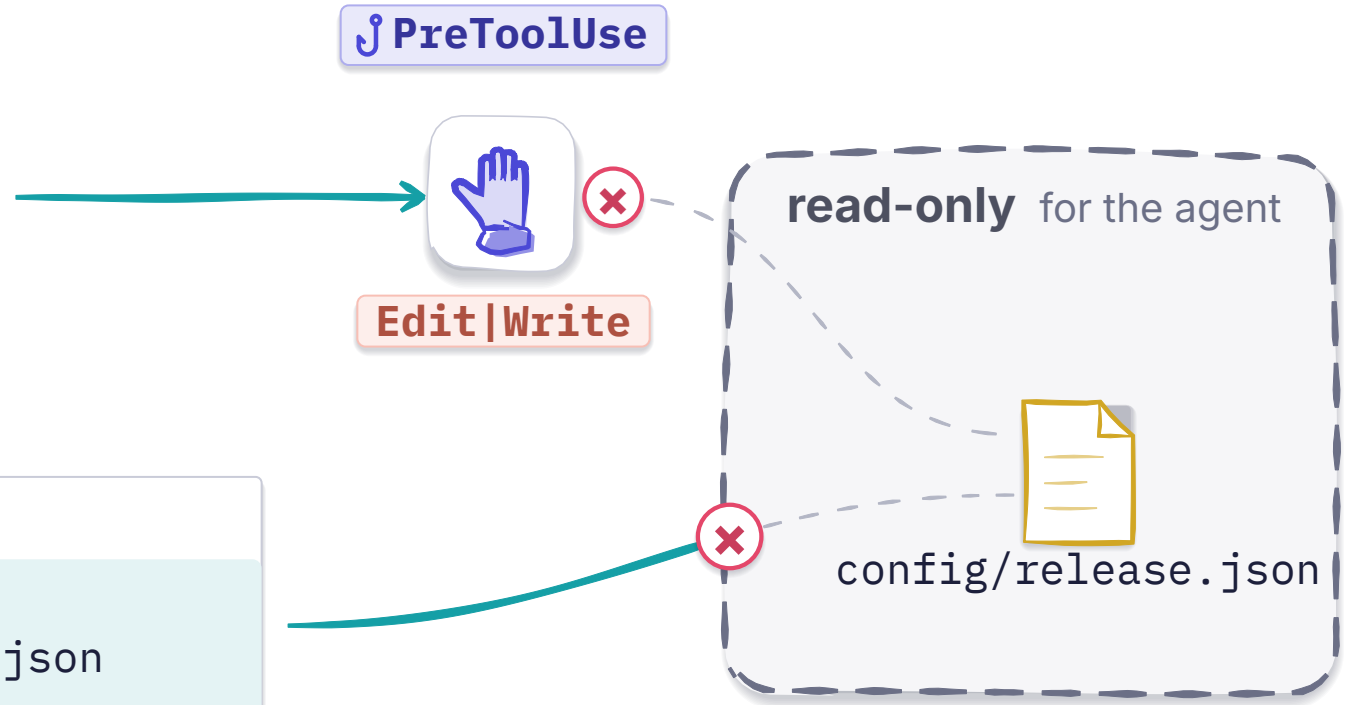
The matcher covers Edit and Write, not Bash

tool call asked by the model

```
Edit config/release.json
```

tool call asked by the model

```
Bash
cat new.json > config/release.json
python tools/bump_version.py
```

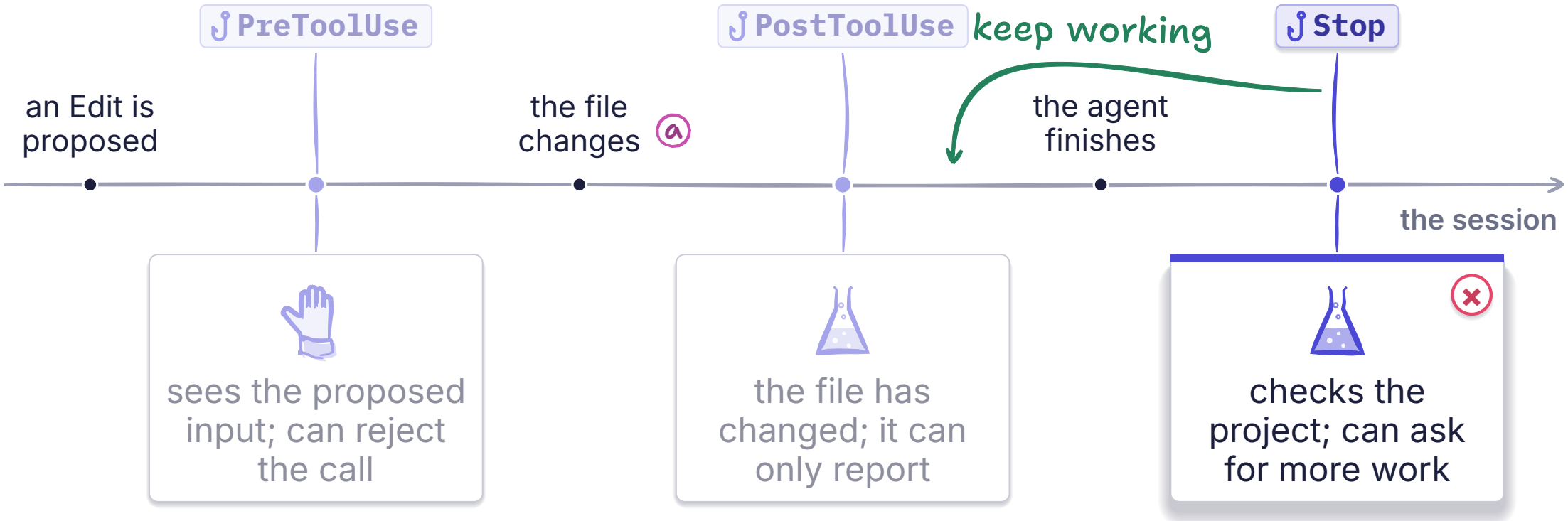


The hook keeps ordinary edits out.

Filesystem permissions or a sandbox: no tool can write it.

Bash commands illustrative

Before the turn ends



@ Earlier edits stay: Stop does not undo them.

A Stop hook checks the project when the agent finishes

inside the harness

Stop

```
check_completion.py
runs bash tools/check.sh
```

timeout 120 s

the project's existing check script, on the current files

Who makes the decision?



a program

the same files give the same answer, with no extra model call



a model's judgment

another model call, and its answer can vary

Use a program when it can make the decision.

The check script stops at the first failure

tools/check.sh

```
1 set -e
2 python -m pytest -q
3 python -m mypy --strict src
4 python -m ruff check .
5 python -m ruff format --check .
```

- the tests
- the type checker
- the linter
- the format check

`set -e`: the first failing command ends the script, and its failure is the result.

A failed check blocks the end of the turn

inside the harness

check passes: the turn can end

Stop

check_completion.py

prints nothing

check fails

Stop

check_completion.py

block

stdout of check_completion.py, on a failure

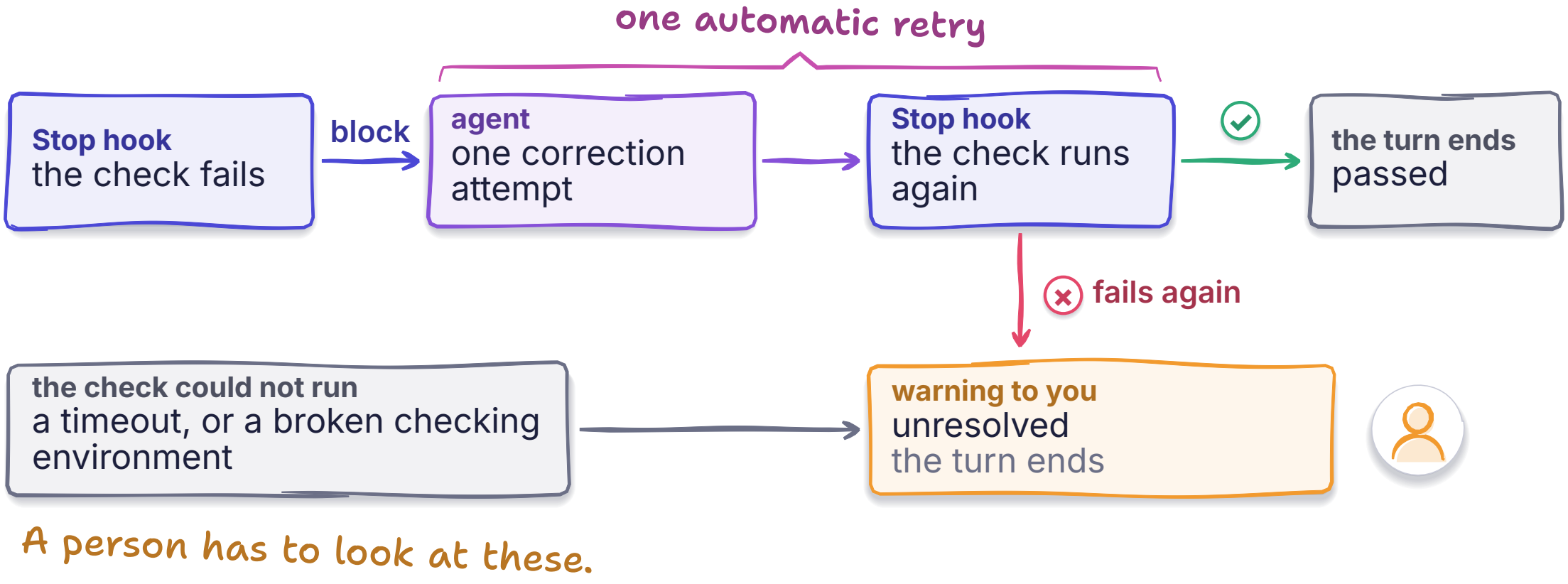
```
{
  "decision": "block",
  "reason": "The project check failed:\n
    E       assert 0 == 5\n
    E        +  where 0 = total_items([2, 3])\n
    FAILED tests/test_counts.py::test_total_it
    Correct the code and run the checks again."
}
```

The failing lines of the test output, copied into the reason.

A different syntax for each event.
PreToolUse: `"permissionDecision": "deny"`
Stop: `"decision": "block"`

handler output on the example project, wrapped and cropped

Allow one correction, then hand the problem to you



A flag in the event says this is a retry

Claude Code → the handler, on standard input

stdin excerpt, on a retry

```
{
  "hook_event_name": "Stop",
  "stop_hook_active": true
}
```

Set when the agent is already continuing after a Stop hook blocked. No counter needed.

the handler's rule

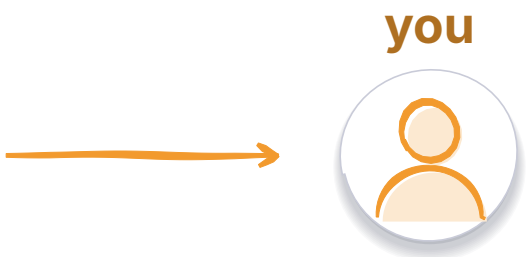
the check	stop_hook_active	
passes	either	prints nothing: the turn ends
fails	`false`	`block`: the agent keeps working
fails again	**`true`**	**warns the user: the turn ends**

It still runs the check again. A second failure warns instead of blocking: that limits the retry.

The warning goes to you, and the turn ends

stdout of `check_completion.py`, on a repeated failure

```
{
  "systemMessage":
    "UNRESOLVED: project checks still fail.
     Automatic retry ended. Inspect
     .claude/check-last.log before
     accepting this work."
}
```



It says the checks still fail, and points to the saved log.

Ending the turn does not mean the work passed.

handler output on the example project, wrapped

See you in the next video.