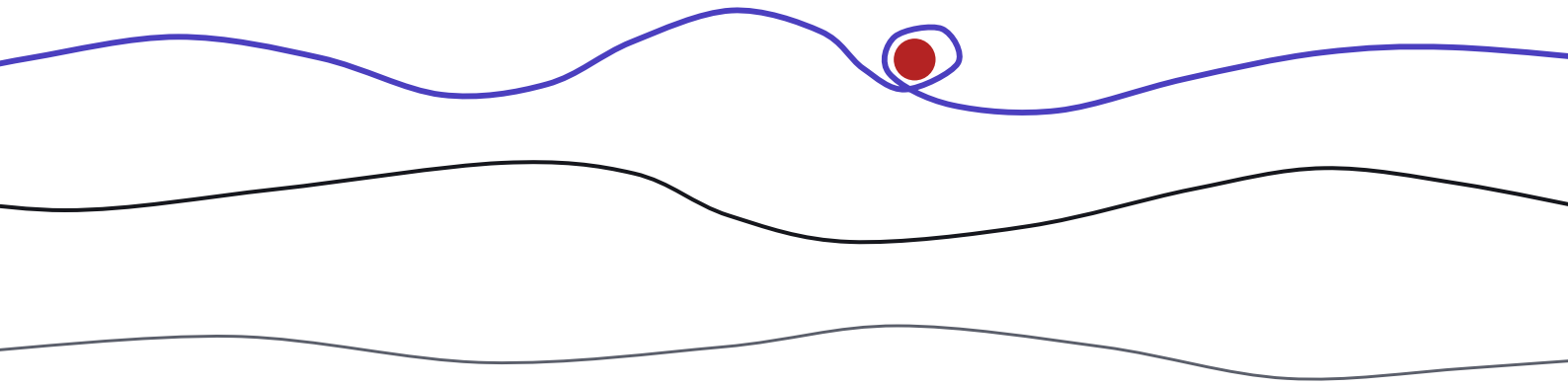
DOCUMENTATION

jupiter.au

OpenUltraCode

An opencode plugin that runs a task as a workflow with acceptance criteria, adversarial review and a completion gate.



VERSION

v2.1.1

DATE

25 November 2026

AUTHOR

belikh

SOURCE

github.com/belikh/OpenUltraCode

Contents

Start

01 **Install** 3

01.1 Requirements 3

01.2 Install 3

01.3 What the installer does 4

01.4 Fusion models 4

01.5 Check the install 4

Guides

02 **Run a workflow** 6

02.1 Start the workflow 6

02.2 What the model does 6

02.3 Read the status 7

02.4 Check the work again 7

02.5 See the record 7

03 **Choose fusion models** 8

03.1 Show the models 8

03.2 Change the models 8

03.3 Fix a model that opencode does not offer 8

04 **Continue a workflow in another session** 9

04.1 Continue the record 9

04.2 Start a new record instead 9

05 **Upgrade and remove** 10

05.1 Upgrade 10

05.2 Recover a replaced file 10

05.3 Remove 10

05.4 Remove an installation from before 2.1.0 11

Reference

06 **Commands** 12

06.1 Workflow commands 12

06.2 Which record a command uses 13

06.3 Fusion options 13
06.4 Configuration command 13

07 **Tools** 14
07.1 open_ultracode_status 14
07.2 open_ultracode_record_intake 14
07.3 open_ultracode_record_finding 15
07.4 open_ultracode_record_verification 15
07.5 open_ultracode_record_blocked_check 16
07.6 open_ultracode_completion_report 16
07.7 open_ultracode_list_models 17
07.8 open_ultracode_configure_fusion 17

08 **Plugin options** 17
08.1 Options 18
08.2 Gate policies 18
08.3 High-effort hints 19
08.4 State directory 19

09 **Workflow record** 19
09.1 Fields 19
09.2 Phases 20
09.3 Evidence that counts 21
09.4 Damaged files 21
09.5 Compaction 21

10 **Installers** 21
10.1 install.sh 22
10.2 install.ps1 22
10.3 install-dev.sh and install-dev.ps1 23
10.4 Checksum 23
10.5 Exit status of install.sh 23
10.6 Runtime check 23

11 **Agents** 24
11.1 Role agents 24
11.2 Fusion agents 25

12 **Glossary** 25

Concepts

13 **Workflow** 27

13.1 Phases 27

13.2 Roles 28

13.3 Size of the workflow 28

13.4 Stop conditions 28

14 **Completion gate** 28

14.1 Rules 29

14.2 Evidence 29

14.3 Findings 29

14.4 End of a workflow 30

14.5 Differences from upstream 2.0.1 30

15 **Session ownership** 30

15.1 What ownership decides 30

15.2 Take over a record 31

15.3 Unknown ancestry 31

15.4 Reason 31

16 **Limits** 31

16.1 Models 32

16.2 Request hints 32

16.3 Reasoning 32

16.4 Enforcement 32

16.5 Notices 32

Develop

17 **Build and test** 33

17.1 Set up 33

17.2 Run the checks 33

17.3 Git hooks 34

17.4 Source layout 34

17.5 Try a change in opencode 35

17.6 Documentation 35

18 **Release** 36

18.1 Prepare 36

18.2 Release with the workflow 37

18.3 When the runtime check fails . . . 37
18.4 Release by hand . . . 37
18.5 Publish the pages . . . 38
18.6 Check the release . . . 38

OpenUltraCode

OpenUltraCode adds nine slash commands, eight tools and ten agents to opencode 2. A command such as `/ultracode` starts a *Workflow*. The model records acceptance criteria, findings and the result of each check in a *Workflow record*. The *Completion gate* then reports one status: `verified`, `partial`, `blocked`, `failed` or `research-only`.

OpenUltraCode keeps the model you selected in opencode. It runs no proxy and adds no model IDs.

This project is a fork of norandom/OpenUltraCode, which created it. The fork starts from upstream 2.0.1. Completion gate and Session ownership describe what it changes.

Start with Install, then Run a workflow.

SECTION

Start

Install it and run it for the first time.

01

Install

Install the latest release into opencode with one command and check that the plugin loaded.

01.1

Requirements

- opencode 2 on your `PATH`.
- Node.js on your `PATH`. The installer scripts run on it. CI tests Node.js 22 and 24.
- Linux and macOS: `curl`, `tar`, and `sha256sum` or `shasum`.

01.2

Install

On Linux and macOS:

INSTALL WITH SH

```
curl -fsSL https://raw.githubusercontent.com/belikh/OpenUltraCode/main/install.sh
| sh
```

On Windows:

INSTALL WITH POWERSHELL

```
irm https://raw.githubusercontent.com/belikh/OpenUltraCode/main/install.ps1 | iex
```

The installer ends with a summary of what it found on disk:

OUTPUT

```
OpenUltraCode 2.1.1 installed
  Source:          /home/you/.local/share/open-ultracode
  Plugin entry:
  /home/you/.local/share/open-ultracode/.opencode/plugins/open-ultracode
  OpenCode config: /home/you/.config/opencode/opencode.json
  Fusion panel A:  mistral/mistral-large-4 (variant max)
  Fusion panel B:  mistral/mistral-large-4 (variant max)
```

```
  Fusion arbiter:  mistral/mistral-large-4 (variant max)
  Commands:        /ultracode-debug /ultracode-fusion-models /ultracode-fusion
  /ultracode-research /ultracode-spec-audit /ultracode-verify /ultracode
  /ultrathink-fusion /ultrathink
  Agents:          10 installed in /home/you/.config/opencode/agents
```

Restart opencode after the install. opencode loads plugins, commands, skills and agents when it starts.

01.3

What the installer does

1. It downloads the latest release archive and its `.sha256` file from GitHub.
2. It compares the archive with the checksum. If they differ, it stops and changes nothing.
3. It unpacks the release to `~/.local/share/open-ultracode`.
4. It asks which model each fusion role uses. In a terminal it shows a menu. Without a terminal, or with `--non-interactive`, it keeps the models of your earlier install. On a first install it uses the release defaults.
5. It copies the skill, the commands and the agents into `~/.config/opencode/`.
6. It adds one plugin entry and one permission rule to `opencode.json`. It leaves every other setting as it is and writes a backup copy named `opencode.json.backup-openultracode-` with a timestamp.
7. It runs `opencode reload` and checks that opencode offers each fusion model.

If an install replaces a skill, command or agent file that you edited, it first saves your version under `~/.config/opencode/open-ultracode-backup/`.

01.4

Fusion models

The fusion commands use three agents with their own models. The release sets all three to `mistral/mistral-large-4` with the `max` variant. To set other models at install time, pass them as flags:

INSTALL WITH CHOSEN FUSION MODELS

```
curl -fsSL https://raw.githubusercontent.com/belikh/OpenUltraCode/main/install.sh
| sh -s -- \
  --panel-a deepseek/deepseek-v4-pro --panel-b zai/glm-5.2 --decider
  moonshotai/kimi-k3
```

If opencode does not offer a model, the installer prints the problem and exits with status 1:

OUTPUT

```
  FAIL ultracode-fusion-panel-a: provider openai is not available in the running
  opencode (openai/gpt-5)
OpenUltraCode: the runtime check found problems (see above). Fix them with
--panel-a/--panel-b/--decider or /ultracode-fusion-models.
```

The files are installed at that point. Run the installer again with models that opencode offers, or follow Choose fusion models.

01.5

Check the install

In opencode, run:

IN OPENCODE

```
/ultracode-verify "Call open_ultracode_status and report the mode."
```

The reply names the mode `verify`. If opencode does not know the command, restart opencode.

SEE ALSO

- Run a workflow
- Installers
- Upgrade and remove

SECTION

Guides

Steps for one task each.

02

Run a workflow

Start a workflow with /ultracode, follow it through review and verification, and read the completion report.

02.1

Start the workflow

In opencode, give the task to a command:

IN OPENCODE

```
/ultracode "Add a --dry-run flag to the deploy script. It prints the plan and changes nothing."
```

The command starts a *Workflow record* for the project and adds this line to the prompt:

OUTPUT

```
OpenUltraCode mode active: comprehensive. The active selected opencode model is preserved.
```

Use `/ultracode-debug` for a failure, `/ultracode-spec-audit` to review a spec, and `/ultracode-research` to attack a plan. Commands lists all nine commands.

02.2

What the model does

1. **Intake.** It writes the goal and the acceptance criteria into the record with `open_ultracode_record_intake`. A criterion that you did not state is marked `inferred`.
2. **Planning and execution.** It makes the change and runs the checks.
3. **Adversarial review.** It looks for defects and records each one with `open_ultracode_record_finding`.
4. **Reconciliation.** It fixes or rejects each finding and records the decision.
5. **Verification.** It runs each check again and records the result with `open_ultracode_record_verification`.
6. **Report.** It calls `open_ultracode_completion_report` and gives you the status.

02.3

Read the status

STATUS	MEANING	WHAT TO DO
`verified`	Every criterion has a passing check, and no finding of high or medium severity is open.	Nothing. The workflow has ended.
`partial`	A criterion has no passing check, or a check was not run.	Ask for the missing checks.
`failed`	The latest result of a check is `fail`.	Ask for a fix, then for the check to be run again.
`blocked`	A check could not run, or a finding of high or medium severity is open.	Remove the blocker or reconcile the finding.
`research-only`	The workflow produced findings or a recommendation.	Nothing. The workflow has ended.

The report lists what stands in the way under `unresolvedRisks`:

COMPLETION REPORT

```
{
  "status": "blocked",
  "summary": "Blocked completion under strict verification policy. open findings:
  F-1.",
  "unresolvedRisks": ["F-1 remains open: The test never exercises the changed
  branch"]
}
```

02.4

Check the work again

To verify a workflow that is still open, run `/ultracode-verify` in the same session:

IN OPENCODE

```
/ultracode-verify "Run every check again and report."
```

`/ultracode-verify` continues the open workflow and keeps its criteria and evidence. Any other command with a different mode starts a new record.

02.5

See the record

Ask for the status at any time:

IN OPENCODE

```
Call open_ultracode_status.
```

The record is the file `.opencode/open-ultracode/state/workflow-state.json` in the project root. The plugin keeps that directory out of git with a `.gitignore` file.

SEE ALSO

- Completion gate
- Tools
- Continue a workflow in another session

03

Choose fusion models

Show and change the models of the two fusion panels and the decider.

`/ultracode-fusion` and `/ultrathink-fusion` send the task to two panel agents and a decider. Each of the three agents has its own model. The model you selected in opencode stays the *Coordinator*.

03.1

Show the models

IN OPENCODE

```
/ultracode-fusion-models
```

The reply lists panel A, panel B and the decider.

03.2

Change the models

Name the roles to change. A role you leave out keeps its model.

IN OPENCODE

```
/ultracode-fusion-models --panel-a deepseek/deepseek-v4-pro --decider
moonshotai/kimi-k3
```

opencode asks for permission before the agent files change. The next fusion run uses the new models.

A reference is `provider/model` or `provider/model#max`. `#max` selects the model's `max` variant and maximum reasoning. The tool refuses `#max` for a model that declares no `max` variant.

To see which references opencode offers, ask for the list:

IN OPENCODE

```
Call open_ultracode_list_models.
```

03.3

Fix a model that opencode does not offer

When a fusion command starts, the plugin checks each fusion model against the models opencode offers. If a provider, a model or a variant is missing, the prompt gets a line that starts with `OpenUltraCode fusion check:`. The model then names the problem, proposes a replacement from the same provider and asks you to confirm it. It changes nothing before you answer.

From a shell, the runtime check prints the same problems:

CHECK THE INSTALLED FUSION MODELS

```
node ~/.local/share/open-ultracode/scripts/runtime-check.mjs --config-dir
~/.config/opencode
```

OUTPUT

```
OpenUltraCode runtime check (static):
```

```
coordinator      opencode default (no model configured)
ultracode-fusion-panel-a  deepseek/deepseek-v4-pro
(/home/you/.config/opencode/agents/ultracode-fusion-panel-a.md)
ultracode-fusion-panel-b  zai/glm-5.2
(/home/you/.config/opencode/agents/ultracode-fusion-panel-b.md)
ultracode-fusion-arbiter  moonshotai/kimi-k3
(/home/you/.config/opencode/agents/ultracode-fusion-arbiter.md)
result: OK
```

Add `--live` to send one small request to each fusion agent. This costs a few model calls.

SEE ALSO

- Commands
- Installers

04

Continue a workflow in another session

Take over a workflow record that a different opencode session started.

A *Workflow record* belongs to the session that ran the command and to the subagents of that session. A tool call from another session gets this answer:

TOOL RESULT

```
{
  "recorded": false,
  "reason": "The workflow record in this project belongs to another session. Start
  a workflow here with an OpenUltraCode command, or continue that record in this
  session by calling open_ultracode_record_intake with adopt: true."
}
```

04.1

Continue the record

1. Open the project in the new session.
2. Ask for the status. The answer has `"ownedByThisSession": false` and shows the goal of the other session's workflow.

   IN OPENCODE

   ```
   Call open_ultracode_status.
   ```

3. If the goal is the work you want to continue, adopt the record:

   IN OPENCODE

   ```
   Call open_ultracode_record_intake with adopt: true.
   ```

   The record now belongs to this session. The earlier session can no longer change it.

04.2

Start a new record instead

Run a command. A command in a new session always starts an empty record and replaces the old one:

IN OPENCODE

```
/ultracode "The new task."
```

SEE ALSO

- Session ownership
- Workflow record

05

Upgrade and remove

Install a newer release over the current one, recover a file the upgrade replaced, and remove OpenUltraCode.

05.1

Upgrade

Run the installer again. It replaces the release and keeps your fusion models.

UPGRADE WITH SH

```
curl -fsSL https://raw.githubusercontent.com/belikh/OpenUltraCode/main/install.sh
| sh
```

UPGRADE WITH POWERSHELL

```
irm https://raw.githubusercontent.com/belikh/OpenUltraCode/main/install.ps1 | iex
```

Restart opencode after the upgrade.

05.2

Recover a replaced file

When an upgrade replaces a skill, command or agent file with different content, it saves the old file first and prints the list:

OUTPUT

```
OpenUltraCode: 2 installed file(s) changed; the previous versions are in
/home/you/.config/opencode/open-ultracode-backup:
  agents/open-ultracode-implementer.md
  skills/open-ultracode/SKILL.md
```

Copy your changes from `open-ultracode-backup/` into the new file. The next install empties the directory before it saves anything, so recover your changes before you upgrade again.

05.3

Remove

REMOVE WITH SH

```
curl -fsSL https://raw.githubusercontent.com/belikh/OpenUltraCode/main/install.sh
| sh -s -- --remove
```

REMOVE WITH POWERSHELL

```
irm https://raw.githubusercontent.com/belikh/OpenUltraCode/main/install.ps1
-OutFile install.ps1
.\install.ps1 -Remove
```

Removal deletes the skill, the nine commands, the ten agents, the release directory and the OpenUltraCode entries in `opencode.json`. It keeps your own agents and commands, every other setting, and `open-ultracode-backup/`.

Workflow records stay in each project under `.opencode/open-ultracode/`. Delete that directory to remove them.

05.4

Remove an installation from before 2.1.0

The installer loads its removal code from the installed release. A release older than 2.1.0 does not have it, and the installer stops with this message:

OUTPUT

```
OpenUltraCode: /home/you/.local/share/open-ultracode was installed by an older
release. Run this installer once without --remove to update it, then try again.
```

Upgrade once, then remove.

SEE ALSO

- Installers

SECTION

Reference

Every command, option and file format.

06

Commands

The nine slash commands, the mode each one starts and the status it ends with.

Each command takes the task as its argument, for example `/ultracode "Add a --dry-run flag"`. A command that gets too little context asks for the missing goal, scope or acceptance criteria before it starts.

06.1

Workflow commands

COMMAND	MODE IN THE RECORD	CHANGES CODE	ENDS WITH
`/ultracode`	`build`	yes	the gate's status
`/ultracode-debug`	`debug`	yes	the gate's status
`/ultracode-verify`	`verify`, or the mode of the open workflow	no	the gate's status
`/ultracode-spec-audit`	`spec-audit`	no	findings, status `research-only`
`/ultracode-research`	`adversarial-research`	no	findings, status `research-only`
`/ultrathink`	`adversarial-research`	no	a recommendation, status `research-only`
`/ultrathink-fusion`	`adversarial-research`	no	a recommendation, status `research-only`
`/ultracode-fusion`	`build`	depends on the task	the gate's status and a fusion trace

`/ultracode` is labelled `comprehensive` in its prompt. The record stores it as `build`.

06.2

Which record a command uses

SITUATION	RESULT
No workflow is open in this session	The command starts an empty record.
A workflow of the same mode is open in this session	The command continues it.
A workflow of another mode is open in this session	The command starts an empty record. `/ultracode-verify` is the exception: it continues the open workflow.
The open workflow belongs to another session	The command starts an empty record.

A workflow is open until its completion report ends it or its phase is set to `completed`. See Completion gate.

06.3

Fusion options

`/ultracode-fusion` and `/ultrathink-fusion` take these options before the task:

OPTION	MEANING	DEFAULT
`--panel <agent>`	A panel agent for this run. Give it twice.	`ultracode-fusion-panel-a` and `ultracode-fusion-panel-b`
`--decider <agent>`	The agent that decides, or `selected-model` for the model you selected.	`ultracode-fusion-arbiter`

A fusion run has five rounds: each panel answers, criticises the other answer, revises its own, ranks the answers, and the decider chooses. A run with one round is reported as `panel-consult`.

06.4

Configuration command

COMMAND	EFFECT
`/ultracode-fusion-models`	Shows the three fusion models.
`/ultracode-fusion-models --panel-a <ref> --panel-b <ref> --decider <ref>`	Changes the named roles. Each option is optional.

A `<ref>` is `provider/model` or `provider/model#max`. This command starts no workflow.

SEE ALSO

- Run a workflow
- Choose fusion models
- Tools

07

Tools

The eight tools the plugin adds to opencode, with their inputs and results.

The model calls these tools during a workflow. Five of them change the *Workflow record*. They work only in the session that started the workflow and in its subagents.

A tool that cannot record answers with `"recorded": false` and a `reason`. A workflow tool that gets wrong input fails with an error that starts with `Invalid input:`. That error does not block the workflow: correct the input and call the tool again.

07.1

open_ultracode_status

Reads the record. It takes no input.

FIELD	MEANING
`active`	`true` while the workflow is open. `false` when no record exists or the workflow has ended.
`ownedByThisSession`	`false` when another session started the workflow.
`mode`, `phase`, `goal`	From the record.
`openFindings`	IDs of findings that are not yet accepted or rejected.
`verification`	The latest result of each check, as `id:result`.
`completion`	The last completion report, if one exists.
`degradations`, `degradationDetails`	Notices about reduced capabilities.

07.2

open_ultracode_record_intake

Sets the goal, the constraints, the assumptions and the acceptance criteria, and moves the phase. Every field is optional.

FIELD	TYPE	MEANING
`goal`	string	The goal a user can observe.
`constraints`	list of strings	Replaces the list.
`assumptions`	list of strings	Replaces the list.
`criteria`	list of `{ id, text, source }`	Adds criteria. A criterion with an existing `id` replaces the old one. `source` is `user`, `spec` or `inferred`. Default: `user`.
`phase`	string	One of `idle`, `intake`, `planning`, `execution`, `research`, `adversarial-review`, `reconciliation`, `verification`, `remediation`, `completed`.

FIELD	TYPE	MEANING
`adopt`	boolean	`true` makes this session the owner of a record that another session started.

Result: `{ "recorded": true, "phase": "...", "criteriaIds": ["AC-1"] }`.

To resume a workflow whose phase is `blocked`, set a phase. To end a workflow, set the phase to `completed`.

07.3

open_ultracode_record_finding

Records a *Finding*, or changes one. To change a finding, send its `id` and the fields that change.

FIELD	TYPE	REQUIRED FOR A NEW FINDING	MEANING
`id`	string	yes	For example `F-1`.
`severity`	`high`, `medium`, `low`	yes	High and medium findings hold the gate while open.
`confidence`	`high`, `medium`, `low`	yes	How sure the reviewer is.
`title`	string	yes	What is wrong.
`evidence`	list of strings	yes	File and line references, command output or quoted requirements.
`impact`	string	yes	What breaks, and for whom.
`recommendation`	string	yes	The fix or the question.
`affectedRequirementIds`	list of strings	no	Criterion IDs.
`disposition`	`accepted`, `rejected`, `clarification-needed`, `deferred`	no	Leave it out while the finding is open.
`dispositionReason`	string	with `rejected`	Why. A new disposition does not keep the old reason.

Result: `{ "recorded": true, "findingId": "F-1", "open": true }`. A finding is open until its disposition is `accepted` or `rejected`.

07.4

open_ultracode_record_verification

Records the result of one check. A second record with the same `id` replaces the first in the gate. The file keeps both.

FIELD	TYPE	REQUIRED	MEANING
`id`	string	yes	Names the check. Use it again when you run the check again.
`kind`	`test`, `build`, `lint`, `typecheck`, `audit`, `manual`, `artifact`	yes	The kind of check.
`result`	`pass`, `fail`, `not-run`, `blocked`	yes	The outcome.
`summary`	string	yes	What was checked.
`criteriaIds`	list of strings	yes	The criteria this check proves. The list can be empty.
`command`	string	no	The command that produced the result.
`artifact`	string	no	A path or URL.
`reason`	string	unless `result` is `pass`	Why the check did not pass.
`residualRisk`	string	unless `result` is `pass`	What stays at risk.

Result: `{ "recorded": true, "evidenceId": "VE-1" }`.

07.5

open_ultracode_record_blocked_check

Records a check that could not run. It takes the fields of `open_ultracode_record_verification` without `result`, and stores the result `blocked`. `reason` and `residualRisk` are required.

07.6

open_ultracode_completion_report

Runs the *Completion gate* over the record, stores the report and returns it.

FIELD	TYPE	MEANING
`researchOnly`	boolean	`true` reports `research-only` in place of `verified`. The research modes set it themselves.

RESULT FIELD	MEANING
`status`	`verified`, `partial`, `blocked`, `failed` or `research-only`.
`summary`	One sentence with the policy and the IDs that hold the gate.

RESULT FIELD	MEANING
`criteria`	Each criterion with the status `verified`, `unverified` or `not-applicable`.
`unresolvedRisks`	What stands in the way, one entry each.
`verificationIds`, `findingIds`, `assumptions`	From the record.

A session that does not own the record gets the report of an empty record, with the status `partial`.

07.7

open_ultracode_list_models

Lists the models opencode offers, grouped by provider. Each entry has the `reference`, its `variants`, and a `maxReference` when the model declares a `max` variant.

FIELD	TYPE	MEANING
`provider`	string	Lists the models of this provider only.

07.8

open_ultracode_configure_fusion

Reads or changes the models of the three fusion agents. With no input it returns the current models. A change asks you for permission.

FIELD	TYPE	MEANING
`panelA`	`provider/model` or `provider/model#max`	The model of fusion panel A.
`panelB`	same	The model of fusion panel B.
`decider`	same	The model of the decider.

The tool refuses a model or a variant that opencode does not offer and names the variants the model declares.

SEE ALSO

- Workflow record
- Completion gate

08

Plugin options

Every option of the plugin entry in opencode.json, with its values and default.

Set options on the plugin entry in `opencode.json`. Restart opencode after a change.

OPENCODE.JSON

```
{
  "plugins": [
    {
      "package":
      "/home/you/.local/share/open-ultracode/.opencode/plugins/open-ultracode",
      "options": {
        "verificationGate": "strict",
        "adversarialReview": "required",
        "highEffort": { "effort": "high", "outputTokens": 32000 }
      }
    }
  ]
}
```

08.1

Options

OPTION	VALUES	DEFAULT	EFFECT
`enabled`	`true`, `false`	`true`	`false` registers no hooks and no tools.
`verificationGate`	`strict`, `advisory`, `disabled`	`strict`	See the next table.
`adversarialReview`	`required`, `recommended`, `disabled`	`required`	What an open finding of high or medium severity does to the report. `required`: `blocked`. `recommended`: `partial`. `disabled`: nothing.
`highEffort. enabled`	`true`, `false`	`true`	Adds request hints to model calls of an open workflow.
`highEffort. effort`	`medium`, `high`, `xhigh`	`xhigh`	The strongest reasoning effort to ask for.
`highEffort. outputTokens`	integer from 1024 to 200000	`64000`	The output token budget to ask for.
`state.directory`	a path inside the project	`.opencode/open-ultracode/state`	Where the workflow record is stored.

A value outside these lists stops opencode from loading the plugin, with an error that names the option:

ERROR

```
Invalid OpenUltraCode plugin options: verificationGate must be one of: strict,
advisory, disabled
```

08.2

Gate policies

VERIFICATIONGATE	A REPORT CAN BE VERIFIED	A PARTIAL REPORT ENDS THE WORKFLOW
`strict`	yes	no
`advisory`	yes	yes
`disabled`	no	yes

08.3

High-effort hints

The hints apply to model calls in the sessions of an open workflow. Other requests in the project are left unchanged.

- `maxTokens` is raised to `highEffort.outputTokens` and capped at the model's output limit.
- `reasoningEffort` is set to the strongest effort the model declares that does not exceed `highEffort.effort`. For a model that declares `low`, `high` and `max`, the setting `xhigh` sends `high`.
- A model that declares no variants, or that opencode does not list, gets no `reasoningEffort`.
- A stronger effort that opencode already set stays.

08.4

State directory

`state.directory` is relative to the project root. The plugin refuses an absolute path, a path with `..` or a backslash, and a path through a symbolic link.

SEE ALSO

- Completion gate
- Workflow record
- Limits

09

Workflow record

The file that holds one workflow, its fields, and when the plugin creates, replaces and ends it.

The plugin stores one *Workflow record* per project:

PATH

```
.opencode/open-ultracode/state/workflow-state.json
```

The path is relative to the project root, also when opencode runs in a subdirectory. The plugin writes a `.gitignore` with the content `*` into the directory, so git ignores the record. It leaves an existing `.gitignore` as it is.

09.1

Fields

WORKFLOW-STATE.JSON

```
{
  "schemaVersion": 1,
  "updatedAt": "2026-11-21T12:57:48.340Z",
  "state": {
    "schemaVersion": 1,
    "sessionId": "ses_e01d0cdf8ffe",
    "mode": "build",
    "phase": "verification",
    "startedAt": "2026-11-21T12:50:02.118Z",
    "updatedAt": "2026-11-21T12:57:48.340Z",
    "goal": "Add a --dry-run flag to the deploy script.",
    "constraints": [],
    "assumptions": [],
    "criteria": [{ "id": "AC-1", "text": "--dry-run changes nothing", "source":
    "user", "status": "pending" }],
    "findings": [],
    "verification": [],
    "degradations": []
  }
}
```


FIELD	WRITTEN BY	MEANING
`sessionId`	a command	The top-level session that owns the workflow.
`mode`	a command	`build`, `debug`, `spec-audit`, `adversarial-research` or `verify`.
`phase`	`open_ultracode_record_intake`, the plugin	See the list below.
`goal`	a command, `open_ultracode_record_intake`	The task text, until intake replaces it.
`constraints`, `assumptions`, `criteria`	`open_ultracode_record_intake`	The intake.
`findings`	`open_ultracode_record_finding`	Every finding with its disposition.
`verification`	the two verification tools	Every recorded check result, oldest first.
`degradations`	the plugin	Notices about reduced capabilities. A new notice replaces an older one with the same ID.
`completion`	`open_ultracode_completion_report`	The last report.

09.2

Phases

`idle`, `intake`, `planning`, `execution`, `research`, `adversarial-review`, `reconciliation`, `verification`, `remediation`, `completed`, `blocked`.

A command sets `intake`. The model moves the phase with `open_ultracode_record_intake`. The plugin sets two phases itself:

PHASE	SET WHEN	LEFT WHEN
`blocked`	You reject a permission request, a rule denies a tool call, or an `open_ultracode_*` tool fails for a reason other than its input.	`open_ultracode_record_intake` sets another phase.
`completed`	A completion report ends the workflow.	The next command starts a new record.

A workflow is open in every phase except `idle` and `completed`.

09.3 Evidence that counts

The `verification` list keeps every record. The gate, the status tool and the compaction summary use the latest record of each `id`.

09.4 Damaged files

The plugin checks the file each time it reads it. A file that is not JSON, has an unknown mode or phase, or lacks one of the lists is moved to `workflow-state.json.bak`. The plugin then reports the notice `workflow-state-corrupt` and continues without a record.

09.5 Compaction

When opencode compacts a session of the workflow, the plugin adds a summary of the record to the compaction prompt. It starts with `OpenUltraCode workflow continuity` and lists the mode, the phase, the goal, the open findings and the latest result of each check. Sessions outside the workflow get no summary.

SEE ALSO

- Tools
- Session ownership

10 Installers

The options and environment variables of the four installer scripts and of the runtime check.

SCRIPT	INSTALLS	SHELL
`install.sh`	the latest release	sh
`install.ps1`	the latest release	PowerShell
`install-dev.sh`	the checkout it is in	sh
`install-dev.ps1`	the checkout it is in	PowerShell

10.1 install.sh

OPTION	EFFECT
`--panel-a <ref>`, `--panel-b <ref>`, `--decider <ref>`	Sets the model of a fusion role. A `<ref>` is `provider/model` or `provider/model#max`.
`--non-interactive`	Shows no menu. Keeps the installed fusion models, or uses the release defaults.
`--archive <file>`	Installs a release archive from disk. If `<file>.sha256` exists, the installer checks the archive against it.
`--verify`	After the install, sends one small request to each fusion agent.
`--skip-runtime-check`	Does not run `opencode reload` or the runtime check.
`--remove`, `-remove`, `-r`	Removes OpenUltraCode.
`--help`, `-h`	Prints the options.

ENVIRONMENT VARIABLE	DEFAULT	MEANING
`OPENULTRACODE_REPO`	`belikh/OpenUltraCode`	The GitHub repository whose latest release is downloaded.
`OPENULTRACODE_HOME`	`~/.local/share/open-ultracode`	Where the release is unpacked.
`OPENCODE_CONFIG_DIR`	`~/.config/opencode`	The opencode configuration directory.

10.2 install.ps1

PARAMETER	SAME AS
`-PanelA`, `-PanelB`, `-Decider`	`--panel-a`, `--panel-b`, `--decider`
`-NonInteractive`	`--non-interactive`
`-Archive`	`--archive`
`-Verify`	`--verify`
`-SkipRuntimeCheck`	`--skip-runtime-check`
`-Remove`	`--remove`
`-Repo`	`OPENULTRACODE_REPO`
`-InstallDir`	`OPENULTRACODE_HOME`
`-OpenCodeConfigDir`	`OPENCODE_CONFIG_DIR`

10.3 install-dev.sh and install-dev.ps1

The development installers take the fusion options, `--non-interactive`, `--skip-runtime-check` and `--remove`, and one more:

OPTION	EFFECT
`--skip-checks` (`-SkipChecks`)	Does not run `pnpm install` and `pnpm run check` before the install.

They register the checkout as the plugin, so a change to the source takes effect after opencode restarts. Their runtime check sends one small request to each fusion agent unless you pass `--skip-runtime-check`. The checks need `pnpm` on your `PATH`.

10.4 Checksum

`install.sh` and `install.ps1` download `open-ultracode-release.tar.gz` and `open-ultracode-release.tar.gz.sha256` from the latest release. When the SHA-256 of the archive differs from the file, the installer stops:

ERROR

```
OpenUltraCode: the release archive does not match its checksum (expected
0000000000000000000000000000000000000000000000000000000000000000, got
3dd90ec3c8805dc55538e7c128803af11134069f9a3649b3f70b148fda1c2cef); the existing
installation was left unchanged.
```

`install.sh` needs `sha256sum` or `shasum` for this.

10.5 Exit status of install.sh

STATUS	MEANING
0	Installed, and the runtime check passed or was skipped.
1	An option is wrong, the download or the checksum failed, or the runtime check found a fusion model that opencode does not offer. In the last case the files are installed.

10.6 Runtime check

RUN THE RUNTIME CHECK

```
node ~/.local/share/open-ultracode/scripts/runtime-check.mjs --config-dir
~/.config/opencode
```

OPTION	EFFECT
`--config-dir <dir>`	The opencode configuration directory to check.
`--live`	Also sends one small request to each fusion agent and checks that it answers on its model.

The check passes when opencode's model list has the provider, the model and the variant of every installed fusion agent. It exits with 1 when it finds a problem and with another status when it cannot reach opencode.

SEE ALSO

- Install
- Upgrade and remove
- Release

11

Agents

The seven role agents and the three fusion agents, with what each one may do.

The installer copies ten agent files into `~/.config/opencode/agents/`. The role agents use the model you selected in opencode. The fusion agents name their own model.

11.1

Role agents

AGENT	ROLE	EDIT	SHELL	OTHER
`open-ultracode`	Coordinator: keeps the phases in order and records intake, findings and evidence.	deny	ask	`task`: ask
`open-ultracode-planner`	Turns the task into steps and a check for each criterion.	deny	ask	
`open-ultracode-implementer`	Makes one bounded change and runs its check.	ask	ask	
`open-ultracode-adversary`	Looks for defects and reports findings.	deny	ask	
`open-ultracode-reconciler`	Gives each finding a disposition.	deny	ask	
`open-ultracode-verifier`	Runs the checks again and reports the results.	deny	ask	

AGENT	ROLE	EDIT	SHELL	OTHER
`open-ultracode-researcher`	Attacks a claim, a spec or a plan.	deny	ask	`webfetch`: ask

The columns show the `permission` block each agent ships with. To stop the questions for an agent, change `ask` to `allow` in its file. An upgrade replaces the file and saves your version in `open-ultracode-backup/`.

Role agents report their results to the session that started them. That session records them with the tools in Tools.

11.2 Fusion agents

AGENT	ROLE	RELEASE DEFAULT
`ultracode-fusion-panel-a`	Panel A	`mistral/mistral-large-4#max`
`ultracode-fusion-panel-b`	Panel B	`mistral/mistral-large-4#max`
`ultracode-fusion-arbiter`	Decider	`mistral/mistral-large-4#max`

A fusion agent may not edit files, run shell commands or change the fusion models. It answers from the context the coordinator gives it.

SEE ALSO

- Choose fusion models
- Workflow

12 Glossary

The terms these pages use for workflows, records, evidence and findings.

Acceptance criterion
A statement that a command, a test or an observation can show to be true or false. Each has an ID such as AC-1.

Completion gate
The rules that turn criteria, evidence and findings into one status: verified, partial, blocked, failed or research-only.

Coordinator
The session that ran the command. It keeps the workflow record and takes the final status from the completion gate.

Evidence
The recorded result of one check: pass, fail, blocked or not-run, with the acceptance criteria it proves.

Finding
A defect or gap that the adversarial review reports, with a severity, a confidence and evidence.

Fusion
A run in which two panel agents on their own models answer, criticise and revise, and a decider chooses the result.

Research mode
The modes spec-audit and adversarial-research. They end in findings or a recommendation and report research-only.

Workflow
One task run through the phases intake, planning, execution, adversarial review, reconciliation and verification. A command such as /ultracode starts it.

Workflow record
The file .opencode/open-ultracode/state/workflow-state.json in the project root. It holds the goal, criteria, findings, evidence and completion report of one workflow.

SECTION

Concepts

How it works and why it is built this way.

13

Workflow

The six phases of a workflow, the roles that do the work, and how much of it a small task needs.

A *Workflow* is one task with a goal, acceptance criteria and a record of what was checked. The `open-ultracode` skill tells the model how to run it. The plugin stores the record and computes the final status.

13.1

Phases

PHASE	WHAT HAPPENS	WHAT IS RECORDED
Intake	The model states the goal, the constraints and the acceptance criteria. It asks you only when the answer changes behaviour, safety or scope.	goal, constraints, assumptions, criteria
Planning	It orders the steps and names the check that proves each criterion.	the phase
Execution	It makes the change. For a change in behaviour it writes the failing check first.	the phase
Adversarial review	It tries to show that the result is wrong.	findings
Reconciliation	It fixes each finding or rejects it with a reason.	dispositions
Verification	It runs each check again and asks the gate for the status.	evidence, completion report

The research modes skip execution. They end after the findings and report `research-only`.

13.2 Roles

Seven agents can do the roles: coordinator, planner, implementer, adversary, reconciler, verifier and researcher. Agents lists them. The coordinator is the session that ran the command. It gives each agent the goal, the criteria and the files in scope, because an agent starts with no memory of the session.

The adversary gets the change, the criteria and the commands to run. It does not get the implementer's own judgement of the work.

When subagents are not available, the model does every role itself in the same session and labels each one. The skill calls this the single-session fallback. The final report then says that the review was not independent.

13.3 Size of the workflow

The skill scales the workflow to the task:

TASK	WHAT RUNS
One file, clear criteria, low risk	The simple-task shortcut: state the criterion, make the smallest change, run one check, report.
Several files or a change in behaviour	Every phase in the main session. The adversary and the verifier run as subagents.
Large, risky or underspecified	Every phase with every role as a subagent, and a spec-audit before any change.

A task that you start with an OpenUltraCode command always runs at least the second row.

13.4 Stop conditions

The skill stops the model in three cases:

- You deny a permission or a tool call. The model stops that path and names the safe next step. It does not look for another route to the same action.
- The same action ran twice with the same result. The model reports it and does not run it a third time.
- A check cannot run. The model records it as blocked once and moves on.

SEE ALSO

- Completion gate
- Run a workflow

14 Completion gate

The rules that turn criteria, evidence and findings into one status, and what ends a workflow.

The *Completion gate* is a function of the *Workflow record*. `open_ultracode_completion_report` runs it. The model reports the gate's status and does not judge completion itself.

14.1

Rules

The gate applies these rules in order and returns the first status that fits:

ORDER	CONDITION	STATUS
1	The latest result of a check is `blocked`.	`blocked`
2	The latest result of a check is `fail`.	`failed`
3	The workflow is in a *Research mode*, or the call sets `researchOnly`.	`research-only`
4	`verificationGate` is `disabled`.	`partial`
5	There is no criterion, a criterion has no passing check, or a check has the result `not-run`.	`partial`
6	A *Finding* of high or medium severity is open.	`blocked` with `adversarialReview: required`, `partial` with `recommended`
7	None of the above.	`verified`

A check proves a criterion when its result is `pass` and its `criteriaIds` contain the criterion's ID.

14.2

Evidence

Each check has an `id`. The gate uses the latest record of each `id`. A check that failed and then passes under the same `id` no longer holds the gate. A check that passed and then fails does.

The record keeps every result, so the history of a check stays readable.

14.3

Findings

A finding is open until it is `accepted` or `rejected`:

DISPOSITION	MEANING	OPEN
none	Reported and not yet decided.	yes
`accepted`	The finding was real and has been addressed.	no
`rejected`	The finding does not hold. The tool requires a reason.	no
`deferred`	Decided for later.	yes
`clarification-needed`	Waiting for an answer.	yes

Findings of low severity never hold the gate. With `adversarialReview: disabled` the report lists findings and they do not change the status.

14.4

End of a workflow

A workflow ends when a completion report has one of these statuses:

STATUS	VERIFICATIONGATE: STRICT	ADVISORY OR DISABLED
`verified`	ends	ends
`research-only`	ends	ends
`partial`	stays open	ends
`failed`, `blocked`	stays open	stays open

Setting the phase to `completed` with `open_ultracode_record_intake` also ends it. After the end, the next command starts an empty record, and the high-effort hints stop.

14.5

Differences from upstream 2.0.1

- The upstream gate did not read findings. A report could be `verified` with an open finding of high severity.
- Evidence could not be replaced. One `fail` record kept the status at `failed` after the check passed.
- Research workflows reported `partial`.
- `advisory` behaved like `strict`.

SEE ALSO

- Tools
- Plugin options

15

Session ownership

Which opencode sessions can change a workflow record, and which events in a project affect the workflow.

A *Workflow record* belongs to one top-level session: the session that ran the command. The sessions that this session starts, such as its subagents, belong to the workflow too. The plugin finds the top-level session of any session by following `parentID` in opencode's session data.

15.1

What ownership decides

EVENT	IN A SESSION OF THE WORKFLOW	IN ANOTHER SESSION OF THE PROJECT
A recording tool is called	The record changes.	The tool answers `"recorded": false` and says why.

EVENT	IN A SESSION OF THE WORKFLOW	IN ANOTHER SESSION OF THE PROJECT
`open_ultracode_completion_report` is called	The report covers the record and is stored.	The report covers an empty record and nothing is stored.
A permission is denied	The phase becomes `blocked`.	Nothing.
A tool fails	A notice is added.	Nothing.
The model is called	The high-effort hints apply.	The request is unchanged.
The session is compacted	The summary of the record is added.	Nothing.
A command is run	The record continues or a new one starts. See Commands.	A new record replaces the old one.

`open_ultracode_status` answers in every session. Its field `ownedByThisSession` says whether the session can change the record.

15.2 Take over a record

A session takes over a record by calling `open_ultracode_record_intake` with `adopt: true`. The record's `sessionId` becomes the top-level session of the caller. Continue a workflow in another session has the steps.

15.3 Unknown ancestry

If opencode cannot report a session's parents, the plugin treats the session as part of the workflow. A lookup that fails cannot lock a workflow out of its own record.

15.4 Reason

One record per project is shared by every opencode session in that project. Before 2.1.0 any session could write to it. Check results of unrelated tasks then collected in one record, and one `blocked` result among them held the gate at `blocked`.

SEE ALSO

- Workflow record
- Completion gate

16 Limits

What OpenUltraCode does not change in opencode, and what its request hints can and cannot do.

16.1

Models

OpenUltraCode uses the model you selected in opencode. It installs no model IDs and no provider routes, and it runs no proxy. Requests go through the providers you configured.

The three fusion agents are the exception. Each names its own model in its agent file, and opencode starts it on that model. You choose those models. See Choose fusion models.

16.2

Request hints

During an open workflow the plugin asks for more output tokens and a higher reasoning effort. Plugin options has the rules. The hints have three limits:

- The plugin sends an effort only when the model declares it. Some providers reject an undeclared value with HTTP 400.
- The plugin caps the token budget at the model's output limit.
- A provider can ignore a hint. The plugin cannot detect that and makes no claim about it.

16.3

Reasoning

OpenUltraCode reads no hidden reasoning and depends on none. The record holds what can be checked: criteria, check results, findings and assumptions.

16.4

Enforcement

The gate reports a status. It does not stop the model from editing files or from saying that the work is done. The skill tells the model to take the status from the gate, and the report is in the record for you to read.

A check result is what the model recorded. The plugin does not run the checks itself.

16.5

Notices

A notice records that something the workflow relies on was reduced or missing. `open_ultracode_status` lists the notices.

NOTICE ID STARTS WITH	CAUSE	BLOCKS THE WORKFLOW
`permission-denied-`	A permission was denied or rejected.	yes
`phase-`	One of the plugin's own tools failed.	yes
`tool-error-`	Another tool failed.	no
`unsupported-`	A fusion model, provider or variant is not available.	no
`workflow-state-corrupt`	The record file could not be read.	no

SEE ALSO

- Completion gate
- Workflow record

SECTION

Develop

Build, test and release the project.

17

Build and test

Set up a checkout, run the checks, install the checkout into opencode and rebuild these pages with the documentation generator.

17.1

Set up

You need Node.js 22 or later and `pnpm` 11.4.0. The repository pins the `pnpm` version for Corepack.

CLONE AND INSTALL

```
git clone https://github.com/belikh/OpenUltraCode.git
cd OpenUltraCode
corepack enable
pnpm install --frozen-lockfile
```

`pnpm install` sets `core.hooksPath` to `.githooks`.

17.2

Run the checks

RUN EVERY CHECK

```
pnpm run check
```

`check` runs five steps and stops at the first that fails:

STEP	COMMAND	WHAT IT CHECKS
Assets	`pnpm run validate:assets`	The skill, the commands, the agents and the workflows have the parts the plugin relies on.

STEP	COMMAND	WHAT IT CHECKS
Installers	`pnpm run validate:installers`	The four installers and their shared files follow the rules in `scripts/validate-installers.ts`.
Types	`pnpm run build`	`tsc --noEmit`.
Lint	`pnpm run lint`	`eslint .`.
Tests	`pnpm run test`	The tests in `tests/` with the Node.js test runner.

The tests install into temporary directories and never touch `~/.config/opencode`. The syntax tests for `dash`, `shellcheck` and PowerShell are skipped when the tool is missing, and the output names each skipped test.

`check` works offline, so it does not run the documentation generator. Documentation has the command for the pages.

17.3 Git hooks

HOOK	WHEN	WHAT IT DOES
`pre-commit`	every commit	Runs `pnpm run check`. It refuses a commit when a staged file has unstaged changes or a project file is untracked, because the checks read the working tree.
`pre-push`	a push of a tag that starts with `v`	Runs `node scripts/runtime-check.mjs --live` against your installed opencode.

17.4 Source layout

PATH	CONTENT
`src/plugin.ts`	The plugin entry: reads the options and registers the hooks.
`src/hooks.ts`	The hooks on prompts, model requests, permissions, tool results and compaction.
`src/tools.ts`	The workflow tools and their input schemas.
`src/gate.ts`	The completion gate.
`src/workflow.ts`	Commands, modes and the rules for starting and continuing a workflow.
`src/sessions.ts`	Finds the top-level session of a session.
`src/state.ts`	Reads and writes the workflow record.

PATH	CONTENT
`src/fusion-*.ts`, `src/model-catalog.ts`	Fusion model configuration and opencode's model list.
`src/opencode-v2.ts`	The part of the opencode 2 plugin API that the plugin uses, as types.
`.opencode/skills`, `.opencode/commands`, `assets/agents`	The skill, the commands and the agents that the installer copies.
`scripts/installer-lib.mjs`	Every edit of `opencode.json` and of installed files.
`scripts/installer-common.sh`, `scripts/installer-common.ps1`	The steps the release and development installers share.
`scripts/docs.mjs`	Fetches and runs the documentation generator.

`tests/plugin-contract.test.ts` compiles `src/opencode-v2.ts` against the real `@opencode/plugin` types, so `pnpm run check` fails when the API changes.

17.5 Try a change in opencode

INSTALL THE CHECKOUT

```
./install-dev.sh --skip-checks --skip-runtime-check
```

The plugin entry in `opencode.json` now points at the checkout. Restart opencode to load a change. Run `./install-dev.sh --remove` to undo it.

17.6 Documentation

These pages are written in `docs-src/` and generated with `jupiter-docs` from belikh/docs. The files in `docs/` are output. Change a page in `docs-src/`, then run:

REBUILD AND CHECK THE PAGES

```
pnpm run docs:build
pnpm run docs:check
```

OUTPUT OF DOCS:CHECK

```
19 page(s) OK
Generated Markdown is current (19 page(s)).
```

Commit `docs-src/` and `docs/` together.

The generator is not on npm, so `pnpm install` cannot provide it. `scripts/docs.mjs` clones it into `.jupiter-docs/` on first use and downloads the design system release that it reads. Git ignores that directory. The first run needs `git`, `npm` and network access.

COMMAND	WHAT IT DOES
`pnpm run docs:check`	Validates the pages and their wording, then fails when `docs/` is out of date.

COMMAND	WHAT IT DOES
`pnpm run docs:build`	Rewrites `docs/` from `docs-src/`.
`node scripts/docs.mjs build --site`	Builds the site into `dist/site`.
`node scripts/docs.mjs --update check`	Fetches the newest generator and design system first, then validates the pages.

Every other argument goes to `jupiter-docs` unchanged.

VARIABLE	DEFAULT	MEANING
`JUPITER_DOCS_DIR`	`.jupiter-docs` in the repository	A checkout of `belikh/docs` to use. The script never changes a checkout that this variable names.
`JUPITER_DESIGN_DIR`	`vendor/design` in the generator checkout	The unpacked design system release.

A checkout that `JUPITER_DOCS_DIR` names must be ready to run. If it is incomplete, the script stops and names each missing part:

ERROR

```
docs: /path/to/docs is missing the generator's dependencies (run `npm ci` in it)
```

The `Docs` workflow runs the validation, the check of `docs/` and the site build on every push to `main` and on every pull request. The release workflow runs it too, because `docs/` is part of the release archive.

SEE ALSO

- Release
- Installers

18

Release

Cut a release that the installers download, through the release workflow or by hand.

The installers download two files from the latest GitHub release: `open-ultracode-release.tar.gz` and `open-ultracode-release.tar.gz.sha256`. A release that is marked as a prerelease is never the latest.

18.1

Prepare

1. Set the new version in `package.json` and in the version check of `tests/assets.test.ts`.
2. Run `pnpm run check` and `pnpm run docs:check`.
3. Merge the change into `main`.

18.2

Release with the workflow

The workflow `.github/workflows/release.yml` runs on a tag of the form `v1.2.3`. GitHub Actions must be enabled for the repository.

TAG AND PUSH

```
git checkout main && git pull
git tag v2.1.1
git push origin v2.1.1
```

Before the push, the `pre-push` hook runs the live runtime check against your installed opencode. The workflow then:

1. runs `pnpm run check` on Linux, macOS and Windows with Node.js 22 and 24
2. validates the pages and fails when `docs/` is out of date
3. fails if the tag differs from the version in `package.json`
4. builds the archive with `scripts/build-release-archive.mjs` and writes its checksum with `sha256sum`
5. creates the release with both files

A tag with a hyphen, such as `v2.2.0-rc.1`, creates a prerelease. A manual run of the workflow creates a prerelease named `manual-` with the run number.

18.3

When the runtime check fails

The check asks the coordinator to call each fusion agent once, through your opencode and your provider accounts. It fails when a model does not answer, whatever the cause: a wrong model name, an account without balance or quota, or a provider that is down. The hook then refuses the push:

OUTPUT

```
pre-push: the runtime check failed; fix the installation before releasing.
```

1. Read the lines that start with `FAIL`. Each names the agent or the provider error.
2. If a model name or a variant is wrong, fix it with Choose fusion models and push again.
3. If your accounts cannot answer, run the same check with models that can. Create a directory with an `opencode.json` that sets `model`, and a project directory with the three fusion agents in `.opencode/agents`, each with a `model` that answers.

RUN THE CHECK WITH OTHER MODELS

```
node scripts/runtime-check.mjs --live --config-dir /tmp/gate/config --directory
/tmp/gate/project
```

LAST LINE OF THE OUTPUT

```
  result: OK
```

`git push --no-verify` skips the hook. The release workflow does not run the runtime check, so nothing else tests the release in a real opencode.

18.4

Release by hand

Use this when Actions is not available.

BUILD, CHECK AND PUBLISH

```
node scripts/build-release-archive.mjs /tmp/release/open-ultracode-release.tar.gz
cd /tmp/release
sha256sum open-ultracode-release.tar.gz > open-ultracode-release.tar.gz.sha256
gh release create v2.1.1 open-ultracode-release.tar.gz
open-ultracode-release.tar.gz.sha256 \
  --repo belikh/OpenUltraCode --title v2.1.1 --target main \
  --notes "OpenUltraCode release archive for install.sh and install.ps1."
```

Run `sha256sum` inside the directory, so the checksum file names the archive without a path.

18.5 Publish the pages

docs.jupiter.au reads the pages from the newest release that has a `docs.config.json`. Its build runs once a day. To publish sooner:

REBUILD THE SITE NOW

```
gh workflow run docs-site.yml --repo belikh/docs
```

18.6 Check the release

Install it into temporary directories. The command fails if the download or the checksum is wrong.

INSTALL THE PUBLISHED RELEASE INTO A SCRATCH DIRECTORY

```
curl -fsSL https://raw.githubusercontent.com/belikh/OpenUltraCode/main/install.sh
|
  OPENCODE_CONFIG_DIR=/tmp/check/config
  OPENULTRACODE_HOME=/tmp/check/open-ultracode \
  sh -s -- --non-interactive --skip-runtime-check
```

OUTPUT

```
OpenUltraCode 2.1.1 installed
```

SEE ALSO

- Build and test
- Installers