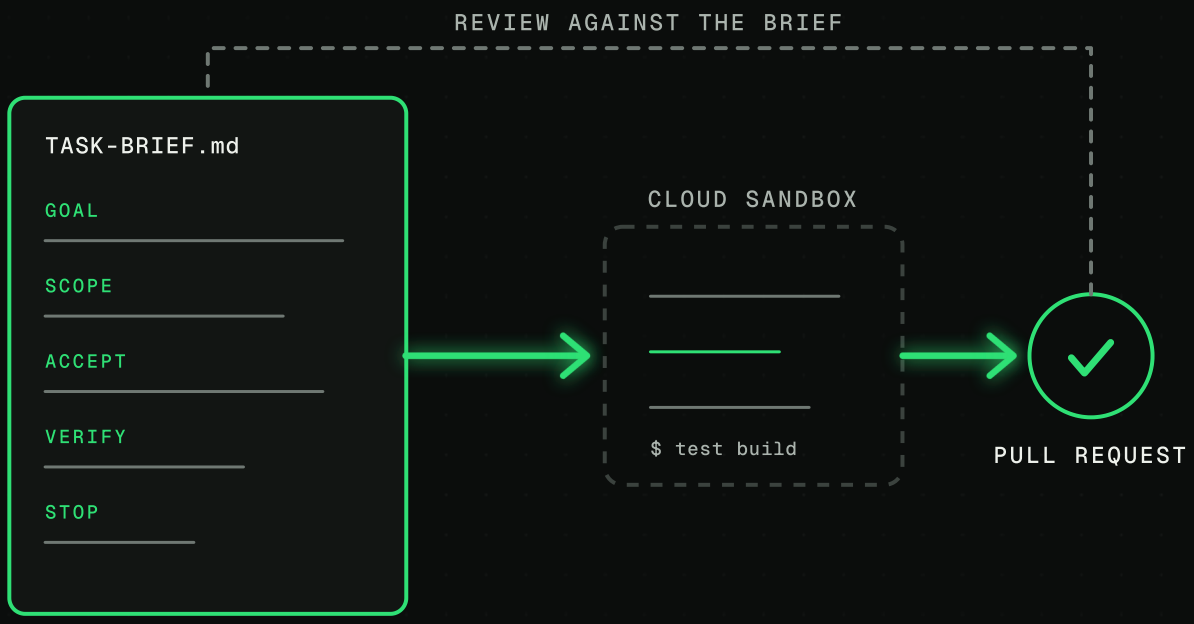


TEMPLATE · FREE

Codex Cloud Task Brief Template

A copy-ready brief for delegating one task to Codex in the cloud, with three filled examples, a setup checklist and a review rubric.

HOW TO USE THIS TEMPLATE

Send Codex cloud tasks that are small, provable and reviewable, and know exactly what sends a returned diff back.

A copy-ready brief for delegating one task to Codex in the cloud, three filled examples, an environment setup checklist and a rubric for reviewing what comes back.

WORKING METHOD

01 Run the setup checklist once per repository, then start one throwaway task to prove the verification commands pass in the cloud.

02 Copy TASK-BRIEF.md, replace every bracket and delete lines that do not apply. Keep standing rules in AGENTS.md, not in the brief.

03 Size the task with the sizing matrix before sending. If it lands in the right-hand column on two rows, split it or keep it local.

04 Paste the finished brief as the task message with Work in set to Cloud and the correct environment selected.

05 When the task returns, follow the runbook and score the diff with the rubric before you open a pull request.

INSIDE

- ✓ Task brief template
- ✓ 3 worked example briefs
- ✓ Cloud environment setup checklist
- ✓ Result review rubric

BEFORE YOU START

A brief is a request, not a control. Protect production with credentials, branch protection and CI, not with a sentence.

A cloud task works from the repositories its environment checks out. Push the starting branch and paste what it cannot open.

Codex Cloud has a current and a legacy experience with different network and secret behavior. Check which one you are configuring.

Saved task state is not source control. Commit or open a pull request for anything you want to keep.

CONTENTS

01	Task brief template	TEMPLATE
02	Worked brief 1: bug fix	TEMPLATE
03	Worked brief 2: small feature	TEMPLATE
04	Worked brief 3: refactor	TEMPLATE
05	Cloud environment setup checklist	24 CHECKS
06	Task sizing: delegate to the cloud or keep local	TABLE
07	Result review rubric	TABLE
08	When the task returns	RUNBOOK

Task brief template

One brief per delegated task. Every section answers a question the reviewer will otherwise have to ask when the diff returns.

TASK-BRIEF.MD · COPY AND ADAPT

```
# Task brief: [SHORT TITLE, VERB FIRST]

Repository: [OWNER/REPO]
Start from: [BRANCH OR COMMIT, ALREADY PUSHED]
Standing rules: follow AGENTS.md in this repository. This brief adds task detail only.
If this brief and AGENTS.md disagree, follow this brief and name the conflict in your report.

## Goal
[ONE OR TWO SENTENCES: WHAT MUST BE TRUE WHEN THIS IS DONE, DESCRIBED AS BEHAVIOR A USER OR CALLER CAN OBSERVE]

## Context and pointers
- Start reading at: [FILE PATHS, MOST RELEVANT FIRST]
- Existing pattern to follow: [PATH TO SIMILAR CODE]
- Current behavior or reproduction: [STEPS, FAILING INPUT, OR LOG LINE, PASTED IN FULL]
- Background the repository does not contain: [PASTED ISSUE TEXT OR DECISION, NOT A LINK THE TASK CANNOT OPEN]

## Scope
In scope:
- [CHANGE 1]
- [CHANGE 2]
Out of scope, do not change:
- [NEIGHBORING CODE, FILES OR BEHAVIOR TO LEAVE ALONE]
- [FOLLOW-UP WORK THAT GETS ITS OWN BRIEF]
Expected size: about [N] files. If the change needs more than [LIMIT] files, stop and report.

## Acceptance criteria
1. [OBSERVABLE RESULT, CHECKABLE BY A TEST OR COMMAND]
2. [OBSERVABLE RESULT FOR THE EDGE CASE: EMPTY, INVALID OR UNAUTHORIZED INPUT]
3. [WHAT MUST STAY EXACTLY AS IT IS TODAY]
4. A test fails on the starting commit and passes on your change: [TEST NAME OR LOCATION]

## Verification commands
Run in this order. Paste the command and the last lines of its output for each.
1. `[FOCUSED TEST COMMAND FOR THE CHANGED AREA]`
2. `[TYPE CHECK COMMAND]`
3. `[LINT COMMAND]`
4. `[BUILD OR FULL TEST COMMAND]`
These must pass inside the cloud environment. Do not skip a command because it is slow.

## Constraints
- Dependencies: [NONE ADDED, REMOVED OR UPGRADED | ALLOWED LIST]
- Data and schema: [NO MIGRATIONS | ONE NEW MIGRATION FILE, NEVER EDIT AN APPLIED ONE]
- Public surface: [ROUTES, EXPORTED TYPES OR CONFIG THAT MUST NOT CHANGE]
- Tests: do not delete, skip or loosen an existing test. If one fails for a reason outside this task, report it.
- Network and credentials: use only what the environment already provides. Do not ask for new access; report the need.
```

```
## What to return
- Summary: what changed and why, in three sentences or fewer.
- Files changed, with one line each, and any file touched that is not listed under In scope.
- Each acceptance criterion with the evidence that proves it (test name or command output).
- Verification output for every command above, marked pass, fail or not run with the reason.
- Assumptions you made, and anything you noticed but deliberately left alone.

## Stop conditions
Stop and report instead of continuing if any of these is true:
- The reproduction does not reproduce, or the cause is outside the files named in Scope.
- An acceptance criterion cannot be met without breaking a constraint.
- A verification command cannot run in the environment (missing tool, service, network or credential).
- The change grows past the expected size, or requires [DECISION RESERVED FOR A HUMAN].
```

01 Write the goal as behavior, not as an implementation. Put the implementation hint, if you have one, under Context so the task can disagree with it.

02 Paste text instead of linking. A cloud task reaches only the destinations its environment allows, so a link to a private tracker may be unreadable.

03 Every acceptance criterion needs a matching line under Verification commands or a named test. A criterion with no check is a wish.

04 Do not restate AGENTS.md. Codex reads AGENTS.md files before doing any work, so repeat a standing rule only when this task changes it.

05 Keep the stop conditions. A task that stops with a clear report is cheaper to review than one that finishes by widening its own scope.

02 / 08 TEMPLATE

Worked brief 1: bug fix

Illustrative example, not a real project. A filled brief for a pagination defect in a generic help desk web app; every path, command and name is invented.

TASK-BRIEF.MD (EXAMPLE: BUG FIX) · COPY AND ADAPT

```
# Task brief: Fix ticket list pagination that repeats a ticket across pages
(ILLUSTRATIVE EXAMPLE. Invented repository, paths and commands for a generic help desk web app.)

Repository: example-org/helpdesk-web
Start from: main at the latest pushed commit
Standing rules: follow AGENTS.md in this repository. This brief adds task detail only.
If this brief and AGENTS.md disagree, follow this brief and name the conflict in your report.

## Goal
Paging through the ticket list shows every ticket exactly once. Today a ticket can appear on two pages
and another can be missing.

## Context and pointers
```

```
- Start reading at: src/server/tickets/listTickets.ts, then src/server/tickets/listTickets.test.ts
- Existing pattern to follow: src/server/customers/listCustomers.ts (pages correctly)
- Current behavior: seed 45 tickets where 30 share the same created_at value, request page 1 and page
2 with a page size of 20, and compare ids. At least one id appears on both pages.
- Background: the list is ordered by created_at only. Tickets imported in bulk share a timestamp. This
is a hypothesis, confirm it before changing code.

## Scope
In scope:
- The ordering used by listTickets so results are stable across pages.
- A regression test that seeds tickets with identical created_at values.
Out of scope, do not change:
- The page size, the response shape, or the cursor and offset parameters.
- The ticket import job, even if it is the source of the shared timestamps.
- The customers list.
Expected size: about 2 files. If the change needs more than 4 files, stop and report.

## Acceptance criteria
1. With 45 seeded tickets, 30 of them sharing one created_at value, pages 1 to 3 at page size 20
contain 45 distinct ids in total.
2. Requesting the same page twice returns the same ids in the same order.
3. For tickets with distinct created_at values, the order is unchanged: newest first.
4. A test fails on the starting commit and passes on your change: add it to
src/server/tickets/listTickets.test.ts.

## Verification commands
Run in this order. Paste the command and the last lines of its output for each.
1. `npm test -- src/server/tickets`
2. `npm run typecheck`
3. `npm run lint`
4. `npm test`
These must pass inside the cloud environment. Do not skip a command because it is slow.

## Constraints
- Dependencies: none added, removed or upgraded.
- Data and schema: no migrations. If a stable order needs a new index, do not add it; describe it in
the report.
- Public surface: the GET /api/tickets response and query parameters must not change.
- Tests: do not delete, skip or loosen an existing test. If one fails for a reason outside this task,
report it.
- Network and credentials: use only what the environment already provides. Do not ask for new access;
report the need.

## What to return
- Summary: the confirmed cause and the fix, in three sentences or fewer.
- Files changed, with one line each, and any file touched outside src/server/tickets.
- Each acceptance criterion with the test name that proves it.
- Verification output for all four commands, marked pass, fail or not run with the reason.
- Assumptions, including whether other list endpoints share the same ordering problem. Do not fix
them.

## Stop conditions
Stop and report instead of continuing if any of these is true:
- The duplicate does not reproduce with the seed described above.
- The cause is in the database layer or the import job rather than listTickets.
- A stable order cannot be achieved without changing the response shape or adding a migration.
- The change grows past 4 files.
```

01 The brief gives a hypothesis and labels it as one. That lets the task confirm the cause first instead of fixing the symptom it was told about.

02 Criterion 3 protects existing behavior. Without it, any stable ordering would pass, including one that reverses the list.

03 The out-of-scope lines name the two most tempting detours: the import job and the sibling list endpoint.

Worked brief 2: small feature

Illustrative example, not a real project. A filled brief for one filter on one list in the same invented help desk app, sized to a single reviewable diff.

TASK-BRIEF.MD (EXAMPLE: SMALL FEATURE) · COPY AND ADAPT

```
# Task brief: Add a status filter to the ticket list
(ILLUSTRATIVE EXAMPLE. Invented repository, paths and commands for a generic help desk web app.)

Repository: example-org/helpdesk-web
Start from: main at the latest pushed commit
Standing rules: follow AGENTS.md in this repository. This brief adds task detail only.
If this brief and AGENTS.md disagree, follow this brief and name the conflict in your report.

## Goal
An agent viewing the ticket list can narrow it to one status (open, pending or closed), and the choice
survives a page reload because it lives in the URL.

## Context and pointers
- Start reading at: src/server/tickets/listTickets.ts, src/web/tickets/TicketList.tsx,
src/shared/ticketStatus.ts
- Existing pattern to follow: the priority filter in src/web/tickets/PriorityFilter.tsx and its
handling in listTickets.ts
- Current behavior: GET /api/tickets ignores unknown query parameters and returns all statuses.
- Background: the allowed statuses are the three values exported from src/shared/ticketStatus.ts. Do
not add a fourth.

## Scope
In scope:
- An optional status query parameter on GET /api/tickets.
- A StatusFilter select on the ticket list, built like PriorityFilter, that reads and writes the
status query parameter.
- Tests for the endpoint and the component.
Out of scope, do not change:
- Combining several statuses in one request.
- Saved views, default filters per user, or any change to the ticket detail page.
- The visual design of the list beyond adding the select beside the priority filter.
Expected size: about 5 files. If the change needs more than 8 files, stop and report.

## Acceptance criteria
1. GET /api/tickets?status=pending returns only tickets whose status is pending, with pagination
working as before.
2. GET /api/tickets?status=archived returns the existing 400 validation error shape and no tickets.
```

```
3. GET /api/tickets with no status parameter returns exactly what it returns today.
4. Choosing a status in the select updates the URL query and reloads the list; choosing All removes
the parameter.
5. Loading the page with ?status=closed in the URL shows the select set to Closed.
6. Tests cover criteria 1, 2, 4 and 5, and the endpoint tests fail on the starting commit.

## Verification commands
Run in this order. Paste the command and the last lines of its output for each.
1. `npm test -- src/server/tickets src/web/tickets`
2. `npm run typecheck`
3. `npm run lint`
4. `npm run build`
These must pass inside the cloud environment. Do not skip a command because it is slow.

## Constraints
- Dependencies: none added. Use the select component already used by PriorityFilter.
- Data and schema: no migrations.
- Public surface: the response shape of GET /api/tickets must not change. The new parameter is
optional.
- Tests: do not delete, skip or loosen an existing test. If one fails for a reason outside this task,
report it.
- Network and credentials: use only what the environment already provides. Do not ask for new access;
report the need.
- Copy: use the labels All, Open, Pending and Closed. Add them where PriorityFilter keeps its labels.

## What to return
- Summary: what was added on the server and in the web app, in three sentences or fewer.
- Files changed, with one line each, and any file touched outside src/server/tickets, src/web/tickets
and src/shared.
- Each acceptance criterion with the test name that proves it.
- Verification output for all four commands, marked pass, fail or not run with the reason.
- Anything a human must check by eye, such as the position of the select at narrow widths.

## Stop conditions
Stop and report instead of continuing if any of these is true:
- PriorityFilter does not use the URL for its state, so there is no pattern to follow.
- The existing validation helper cannot express an enum parameter without modification.
- Filtering by status requires a schema or index change.
- The change grows past 8 files.
```

01 The feature is cut to one parameter and one control. Multi-select and saved views are named as out of scope so they are not built on the way.

02 Criterion 2 pins the error behavior to an existing shape. The task does not get to invent a new error format.

03 The return list asks for what a human must check by eye. A cloud task can run tests; do not assume it has judged layout unless it shows evidence.

Worked brief 3: refactor

Illustrative example, not a real project. A filled brief for a behavior-preserving refactor in the same invented app, where the proof is that nothing observable changes.

TASK-BRIEF.MD (EXAMPLE: REFACTOR) · COPY AND ADAPT

```
# Task brief: Replace three copies of relative-time formatting with one module
(ILLUSTRATIVE EXAMPLE. Invented repository, paths and commands for a generic help desk web app.)

Repository: example-org/helpdesk-web
Start from: main at the latest pushed commit
Standing rules: follow AGENTS.md in this repository. This brief adds task detail only.
If this brief and AGENTS.md disagree, follow this brief and name the conflict in your report.

## Goal
Relative times such as "5 minutes ago" are produced by one function in one module. Output for every
existing caller is identical to today.

## Context and pointers
- Start reading at: src/web/tickets/formatAge.ts, src/web/comments/timeAgo.ts,
src/web/activity/relativeTime.ts
- Existing pattern to follow: shared web helpers live in src/web/lib with a test file beside each.
- Current behavior: three functions implement the same thresholds with small differences. Find and
list the differences before changing anything.
- Background: this is preparation for localizing these strings later. Localization is not part of this
task.

## Scope
In scope:
- A new src/web/lib/relativeTime.ts with tests.
- Updating the callers of the three existing functions to use it.
- Deleting the three old files once nothing imports them.
Out of scope, do not change:
- The wording, thresholds or rounding of any output, even where the three copies disagree. Preserve
each caller's current output through options.
- Absolute date formatting anywhere in the app.
- Any server code.
Expected size: about 12 files, mostly one-line import changes. If the change needs more than 20 files,
stop and report.

## Acceptance criteria
1. Before editing callers, add characterization tests that record the current output of each of the
three functions at these ages: 0 seconds, 59 seconds, 60 seconds, 59 minutes, 60 minutes, 23 hours, 24
hours, 6 days, 7 days, 400 days, and a time in the future.
2. Those characterization tests pass unchanged against the new module through each caller's entry
point.
3. A search for formatAge, timeAgo and the old relativeTime import path returns no matches outside the
new module.
4. No snapshot or existing test expectation is edited.

## Verification commands
Run in this order. Paste the command and the last lines of its output for each.
1. `npm test -- src/web/lib/relativeTime`
2. `npm test -- src/web`
3. `npm run typecheck`
```

```
4. `npm run lint`
5. `npm run build`
These must pass inside the cloud environment. Do not skip a command because it is slow.

## Constraints
- Dependencies: none added. Do not introduce a date library.
- Data and schema: not applicable; no server or database files change.
- Public surface: component props and rendered text must not change.
- Tests: do not delete, skip or loosen an existing test. Tests for the three deleted files move to the new module with their expectations intact.
- Commits: make the characterization tests one commit and the refactor a second commit, so a reviewer can run the tests against the old code.

## What to return
- Summary: what was consolidated, in three sentences or fewer.
- The list of behavioral differences found between the three copies, and the option that preserves each.
- Files changed, grouped as new, edited (import only), edited (more than import) and deleted.
- Verification output for all five commands, marked pass, fail or not run with the reason.
- Any caller whose output you could not prove unchanged.

## Stop conditions
Stop and report instead of continuing if any of these is true:
- A caller outside src/web imports one of the three functions.
- Two copies differ in a way that cannot be preserved with an option, so one caller's output would have to change.
- An existing snapshot would need updating.
- The change grows past 20 files.
```

01 A refactor brief defines done as no observable change, so the first criterion is to record current behavior before touching it.

02 The two-commit constraint is for the reviewer: the first commit can be checked out alone to confirm the tests pass against the old code.

03 Where the copies disagree, the brief says to preserve the disagreement. Deciding which copy is right is a product decision for a separate task.

05 / 08 CHECKLIST

Cloud environment setup checklist

Twenty-four checks to run once per repository before you delegate. UI labels follow the OpenAI Codex documentation as of the date in the research notes; confirm any label that has moved.

Repository and environment

4 CHECKS

- [ ] 01 Start a new task, choose Work in then Cloud, and open Select environment; expect the default environment or a published environment that lists your repository.

- [ ] 02 Create the environment from Select environment or from Settings, Codex Cloud, Environments; select only the repositories this work needs and connect GitHub when prompted.
- [ ] 03 Confirm the repository host is supported: the docs cover GitHub.com and GitLab.com connections and list self-hosted GitHub Enterprise Server as not currently supported.
- [ ] 04 Open the environment's Who can use setting; expect Only me for a private setup, or a named audience you chose on purpose for a shared one.

Setup and saved state

4 CHECKS

- [ ] 05 Select Get started and let Codex inspect the repository, install dependencies and test the workflow; answer its questions about versions, commands and services instead of dismissing them.
- [ ] 06 Read the setup report, the Install script and the Start skill before publishing; expect the install and service start steps to match the repository's own setup instructions.
- [ ] 07 Select Publish, wait for Environment published, then start one throwaway task that runs your test command; expect a pass with dependencies already in place.
- [ ] 08 After a dependency or tooling change, open Edit from the environment menu and select Republish; start a new task, because existing tasks keep their own earlier state.

Network access

4 CHECKS

- [ ] 09 Open the environment's service connection settings and record the internet access state you see; the current guide does not state a default, so do not assume one.
- [ ] 10 Leave Enable all public internet access off unless a task needs it; the docs say turning it on disables the narrower package-manager and additional-domain controls.
- [ ] 11 Turn on Enable package manager domains only if tasks install packages during a run; prove it by running an install command in a throwaway task.
- [ ] 12 Add each further host under Additional allowed domains by exact hostname; per the docs a root domain also allows www, and other subdomains need their own entries.

Variables and secrets

4 CHECKS

- [ ] 13 List every value your verification commands read; store values a program must read directly as environment variables and treat them as visible to the task.
- [ ] 14 Store a credential that is sent to one HTTPS service as a network secret; expect programs to receive a placeholder that the proxy replaces for allowed destinations.
- [ ] 15 After saving an environment-owned network secret, reopen the allowed destinations; expect that secret's host to have been added to restricted internet access.

- [ ] 16 Review each stored value and confirm it is a test or read-only credential; expect no production database URL, deploy token or signing key among environment variables.

Instructions and return path

4 CHECKS

- [ ] 17 Commit AGENTS.md with exact lint, test and build commands, then ask a throwaway cloud task to list the instructions it loaded; expect your commands in the answer.
- [ ] 18 Commit any skill the task depends on to the repository; per the docs, repository skills are available in cloud tasks and personal skills on your computer are not.
- [ ] 19 Push the branch a task should start from and ask the task to print its starting commit; expect the hash to match your pushed commit.
- [ ] 20 Take one throwaway task through review to a pull request; confirm the target branch is correct and that branch protection and CI ran on it.

If the environment is Codex Cloud (Legacy)

4 CHECKS

- [ ] 21 Check which guide matches your screen; the docs label the older environments page Codex Cloud (Legacy), kept for Code Review and the Linear and GitHub integrations.
- [ ] 22 In a legacy environment, read the agent internet access setting; the docs state it is off by default in the agent phase while setup scripts have internet access.
- [ ] 23 In a legacy environment, move values the agent needs out of secrets; the docs state secrets reach setup scripts only and are removed before the agent phase.
- [ ] 24 If you turn legacy agent internet access on, choose the narrowest domain allowlist and restrict HTTP methods to GET, HEAD and OPTIONS; confirm a POST is refused.

06 / 08

REFERENCE

Task sizing: delegate to the cloud or keep local

Check the task against each row before you write the brief. A task that lands in the right-hand column on two or more rows should be split or kept local.

Splitting usually means one brief for the mechanical part and a local session for the part that needs judgment. Send the mechanical part first so its diff is merged before the judgment work starts.

SIGNAL	GOOD CLOUD TASK	KEEP LOCAL OR SPLIT FIRST
Definition of done	Expressible as tests or commands that pass or fail	Depends on taste, a design review or a product decision
Reach of the change	One module or one vertical slice; a reviewer can read the diff in one sitting	Crosses several services, or the file list cannot be estimated

SIGNAL	GOOD CLOUD TASK	KEEP LOCAL OR SPLIT FIRST
Where the proof runs	Unit and integration tests that run in the environment	Needs a real device, a production-like dataset or a manual click-through
Access needed	Repository plus package registries and test credentials	Needs production data, internal systems or a credential you would not store
Starting point	A pushed branch or commit	Uncommitted work on your computer that you are still changing
Cost of a wrong answer	Caught by tests and review before merge	Irreversible: applied migrations, data backfills, published packages
Feedback loop	You can wait and review once at the end	You expect to steer every few minutes while exploring

07 / 08 REFERENCE

Result review rubric

Score the returned task against each row. Any row in the third column sends the task back with a follow-up message; it does not get fixed silently in review.

Use the rubric as written criteria in your follow-up. A message that names the failed row and the evidence expected is itself a small brief.

CRITERION	PASS LOOKS LIKE	SENDS IT BACK	WHERE TO LOOK
Scope	Changed files match the In scope list; extras are explained	Unrelated renames, reformatting or files from the Out of scope list	File list in the diff
Acceptance criteria	Each criterion is mapped to a named test or command output	A criterion is restated as done with no evidence	The task's report
Regression test	A new test fails on the starting commit and passes on the change	No new test, or a test that passes on both commits	Run the test on the base branch
Verification commands	Every listed command shown with output and a pass	A command skipped, replaced or summarized without output	Report, then your own run
Existing tests	No test deleted, skipped or loosened; snapshots untouched unless requested	Edited expectations or new skip markers	Search the diff for test files
Constraints	No dependency, schema or public surface change beyond the brief	A lockfile, migration or exported type changed without being asked	Manifest, lockfile, migrations directory
Secrets and access	No credentials, tokens or internal hostnames in code, tests or logs	A literal secret, or new network calls to unlisted hosts	Diff search and the work log
Honesty of the report	Assumptions and untested areas are listed	Claims about code the diff does not touch, or no stated risk	Report against the diff

CRITERION	PASS LOOKS LIKE	SENDS IT BACK	WHERE TO LOOK
Mergeability	Applies cleanly to the current base; CI passes on the pull request	Conflicts with the base, or CI fails on a check the report called passing	Pull request checks

When the task returns

Seven steps from a finished cloud task to a merged pull request or a clear decision to discard it.

A follow-up in the same chat continues that task's own saved files. Use a new task when the brief itself was wrong, so the next attempt starts from a clean checkout rather than from the earlier attempt's changes.

01 Read the task's report before the diff. If it does not list verification output for every command in the brief, send a follow-up asking for the missing output and stop here.

02 Compare the changed file list with the In scope list. If any file is outside it and unexplained, ask in the same chat for it to be reverted or justified.

03 Read the test changes first, then the implementation. If a new test would pass without the implementation, send it back under the Regression test row.

04 Score the remaining rubric rows. If two or more rows fail, ask for the changes in one follow-up message that names each row and the evidence you expect.

05 If a follow-up round does not clear the same row, stop iterating. Rewrite the brief with narrower scope or more context and start a new task.

06 When every row passes, open a pull request from the task. Wait for CI on the pull request; a pass inside the task environment does not replace it.

07 Run the focused test command yourself from the pull request branch before approving. If your result differs from the report, treat the environment as misconfigured and rerun the setup checklist.

FIELD NOTES

Common mistakes, useful habits, hard limits.

01 **Sending a goal with no way to prove it**

A brief that says make search faster or clean up the settings page gives the task nothing to run and gives you nothing to check. Write the acceptance criteria and the commands first. If you cannot, the task is not ready to delegate.

02 **Assuming the cloud task can see what you can see**

The task runs in its own workspace from the repositories its environment checks out, with the network destinations and values that environment allows. Unpushed commits, a ticket behind a login and a service on your laptop are not part of that. Push the branch and paste the text.

03 **Fixing a failed result by hand in review**

Quietly repairing a returned diff hides the gap in the brief or the environment, and the next task fails the same way. Send it back with the rubric row that failed, then update the brief template or AGENTS.md so the fix is permanent.

PRACTICAL TIPS

Write the What to return section before the Goal. Knowing what you will check at the end usually shrinks the task to a size that can be checked.

Keep one brief per pull request. If the brief needs the word and in its title, it is probably two briefs.

Move anything you write in three briefs in a row into AGENTS.md. The brief should carry only what is specific to this task.

In a shared environment, each person supplies requested values from their own personal vault. Per the docs, sharing an environment shares the requirements, not your credentials.

Save finished briefs next to the pull requests they produced. A brief that returned a clean diff is the best starting point for the next similar task.

WHEN NOT TO USE THIS

For exploratory work where you do not yet know what done looks like. Investigate locally or ask a question first, then write the brief from what you learn.

For changes that touch production data or need production credentials. Do not store those in a cloud environment to make a task possible.

As a replacement for AGENTS.md or for prompts you reuse. Standing rules belong in AGENTS.md; this brief describes one task once.

SOURCES · CHECKED 2026-11-22

OpenAI Codex docs: Codex Cloud · Cloud tasks are started on the web or desktop by choosing Work in then Cloud and picking an environment (the default one or a published environment); Codex can check out repositories available through a connected GitHub or GitLab account; each task has its own workspace and can keep working while your computer is asleep; you inspect changed files and check results, request follow-up changes in the same chat, and commit or open a pull request when ready. Reached by a 308 redirect from https://developers.openai.com/codex/cloud.
https://learn.chatgpt.com/docs/cloud

OpenAI Codex docs: Cloud environments · Environments are created from Select environment then Create environment, or Settings, Codex Cloud, Environments; you select repositories and connect GitHub if prompted; Get started has Codex inspect the repositories, install dependencies and tools and test the workflow; the tested setup can be recorded as an Install script and a Start skill; you review the setup report and select Publish, and Environment published appears; new tasks start from the published prepared filesystem while existing tasks keep their own saved files; updates go through Edit and Republish and apply to new tasks only; saved state does not replace source control and task state is recoverable for up to seven days by default; Enable all public internet access disables the narrower package manager and Additional allowed domains controls; for root domains the www hostname is also allowed and other subdomains need their own entries; allowing a destination does not supply credentials; environment variables are passed directly to programs; network secrets give programs a placeholder that the proxy substitutes for allowed HTTPS destinations; saving environment-owned network secrets adds their destinations to restricted internet access; Who can use can be set to Only me; sharing shares requirements, not credentials, with values supplied from each person's personal vault; repository skills are available in cloud tasks and personal local skills are not synced; self-hosted GitHub Enterprise Server and computer use are not currently supported. The page does not state a default internet access setting and does not mention AGENTS.md.
https://learn.chatgpt.com/docs/environments/cloud-environments

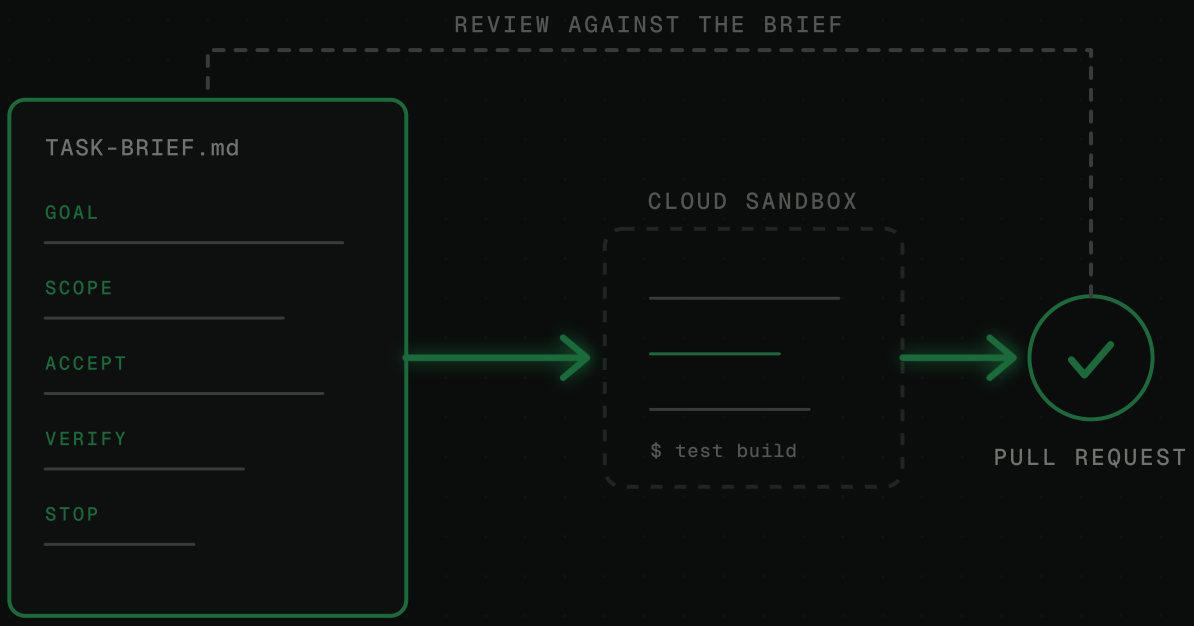
---

OpenAI Codex docs: Cloud environment, Codex Cloud (Legacy) · Codex Cloud (Legacy) continues to support environments for Code Review and the Linear and GitHub integrations. In a legacy environment Codex creates a container, checks out the repo at the selected branch or commit, and runs the setup script; setup scripts run with internet access; agent internet access is off by default; if the repo has AGENTS.md the agent uses it to find project-specific lint and test commands; when the agent finishes it shows its answer and a diff; environment variables are available in the setup and agent phases; secrets are available only to setup scripts and are removed before the agent phase. Reached by a 308 redirect from https://developers.openai.com/codex/cloud/environments.
https://learn.chatgpt.com/docs/environments/cloud-environment

---

OpenAI Codex docs: Agent internet access, Codex Cloud (Legacy) · By default Codex blocks internet access during the agent phase while setup scripts keep it; access is configured per environment as Off or On; domain allowlist presets are None, Common dependencies and All (unrestricted); allowed HTTP methods can be restricted to GET, HEAD and OPTIONS; listed risks are prompt injection from untrusted web content, exfiltration of code or secrets, malware or vulnerable dependencies, and license-restricted content. Reached by a 308 redirect from https://developers.openai.com/codex/cloud/internet-access.
https://learn.chatgpt.com/docs/cloud/internet-access

OpenAI Codex docs: Custom instructions with AGENTS.md · Codex reads AGENTS.md files before doing any work; instructions are combined from the project root down to the current directory, and files closer to the current directory take precedence. The page does not discuss cloud tasks specifically.
https://learn.chatgpt.com/docs/agent-configuration/agents-md



FAISAL KARKOH / RESOURCES

More working files for building with AI.

Prompt packs, checklists, templates and playbooks for shipping reliable AI products. Free to download and reuse in your own work.

LIBRARY
faisalkarkoh.com/resources

WRITING
faisalkarkoh.com/blog

WORK TOGETHER
faisalkarkoh.com/contact