Transactions prevent; convergence repairs

Key takeaways

1. No transaction spans your process and a service; every effect opens a window.
2. Stable identity makes retry converge: search, verify, create only if absent.
3. Recovery observes the service and reconciles; it never replays the plan.

Common misunderstandings of service retries

Which claim would you defend right now? Choose before the evidence.

“A failed client request means the service made no change.”

“Retrying the same create call is safe when the payload is unchanged.”

“A durable local receipt is enough to decide what exists remotely.”

Keep your choice. Look for the mechanism or counterexample that would make it fail.

Preserving application invariants when networked services fail

The invariant: one task, one pull request — and no transaction protects it

What the client saw

```
$ gh pr create \
    --title 'Handle empty cells'
POST /repos/course/project/pulls
connect: network timeout

$ echo $?
1
```

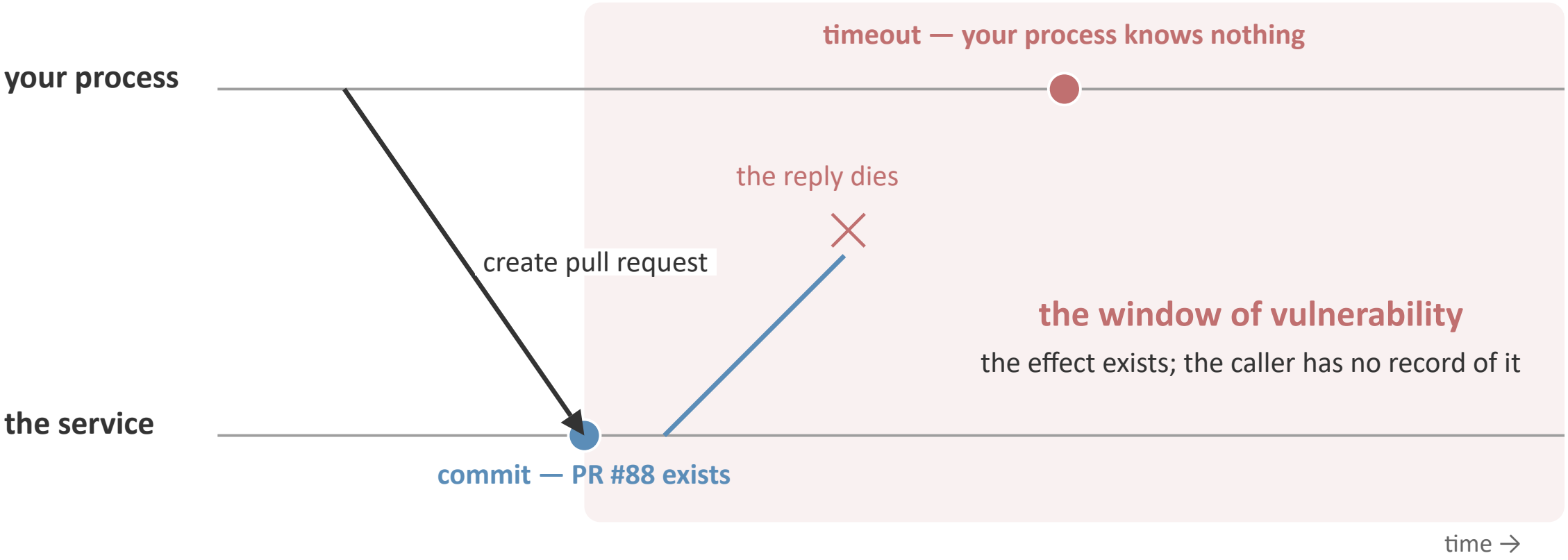
What the server did

```
201 Created
{
  "number": 88,
  "title": "Handle empty cells",
  "head": {"sha": "6a1d9c4"}
}
# the reply died on the way back
```

- Retry blindly and a second pull request breaks the invariant.
- Give up blindly and the effect exists with no record — also broken.

The window of vulnerability

Opened by the server's commit; closed only by your record of it



Only observing the service closes the window — retrying and giving up are both blind guesses.

No transaction spans your process and the service

Exactly-once is impossible; recovery must survive either wrong guess

Ordering	A crash leaves	Blind recovery then
record intent, then create	a record of a PR that may not exist	skips the missing effect
create, then record	a real PR with no local record	creates a duplicate
one transaction around both	impossible: the service cannot join your commit	—

- The window between the server’s commit and your record cannot be closed, only made safe.
- Safe means: on restart, observe the service and converge — do not replay the plan.

Give each external effect a stable identity

Look up, verify exactly, create only when absent

```
marker = "task:issue-412@92f3a10"

existing = github.find_pull_request(marker=marker)
if existing is None:
    pr = github.create_pull_request(
        head="6a1d9c4", body=f"<!-- {marker} -->"
    )
else:
    require(existing.head_sha == "6a1d9c4")
    pr = existing
record(marker, pr.number)
```

- Rerunning this block converges on one pull request no matter where the last attempt died.
- The marker names the task and the exact content: the issue plus the commit being proposed.

Adopt only what verifies exactly

Adoption by resemblance

```
found = github.search_pulls(
    title="Handle empty cells"
)
if found:
    record(found[0].number)

# a human's draft, another task's
# PR, or last week's abandoned
# attempt all match this title
```

Adoption by identity

```
found = github.find_pull_request(
    marker=marker
)
if found is not None:
    require(found.head_sha == "6a1d9c4")
    require(found.base == "main")
    pr = found            # adopt
else:
    pr = create_pull_request(...)
```

- Names collide; embedded identities do not. Match on what you wrote, not on what looks right.
- A mismatch is a stop condition — the world changed in a way the plan did not predict. Surface it.

The same window of vulnerability is everywhere

External effect	Blind retry produces	Stable identity
create a pull request	a duplicate PR to triage and close	marker in body + head SHA
charge a card	a double charge and a refund	Idempotency-Key header (Stripe)
send an email	a duplicate message to everyone	your own Message-ID
publish a package version	an error, not a duplicate	name@version, immutable

When the service offers native identity — idempotency keys, immutable versions — use it. Otherwise embed a marker you can search for.

source: docs.stripe.com/api/idempotent-requests

Keep a durable receipt for every settled effect

The last write of a settled effect — and the first read of the next run

```
{
  "job_id": "task-412",
  "marker": "task:issue-412@92f3a10",
  "effect": {"kind": "pull_request", "number": 88,
             "head_sha": "6a1d9c4"},
  "outcome": "adopted",
  "verified": {"head_sha": "match", "base": "match"},
  "request_id": "req_01...",
  "recorded_at": "2026-08-07T15:21:04Z"
}
```

- Unsynced state dies with the process: write, fsync, rename — or the crash that lost the reply also erases the evidence.
- A receipt is a hint, not the truth: recovery still observes the service before acting on it.

Transactions prevent violations; convergence repairs them

1. No transaction spans your process and a service, so every remote effect opens a window of vulnerability: an effect with no record, or a record with no effect.
2. Retry preserves the invariant only when the effect has a stable identity: search, verify exactly, create only when absent — that block converges from any crash point.
3. The record must be durable — write, fsync, rename — and still ranks below the service: recovery observes the world and reconciles; it never replays the plan.

Recover an unknown-outcome write

10 min

- A client posts a build report as a GitHub issue comment containing `<!-- build:repo@sha -->`. The POST times out; the client crashes before saving the comment ID.
- On restart, search finds: (A) zero markers, (B) one exact marker and body, or (C) one marker with a different body.
- In pairs, write the action for each case and the condition that stops for a human. Another pair challenges one unsafe retry or adoption.
- Deliverable: a three-row recovery table plus the revised stop rule.