# Start with the failed expectation

**Caller**

expects a useful response

→

**Observation**

502 Bad Gateway

→

**Question**

which hop failed?

A symptom is a user-impact claim to bound, not a root cause to announce.

# Separate observation from explanation

## The API process is listed as running and the caller sees 502. Which claim is still untested?

> The proxy may still target the wrong service or port; the API may be unready; and the route may not match the declared topology. Running is one process fact, not a completed request.

Process state is one observation inside a larger request path.

# Choose a small incident envelope

IMPACT

## who · what · when

UTC start and expected behavior

BOUNDARY

## project · services

one time window and owner

A useful incident starts with impact, time, scope and an ownerable boundary.

# Compose is a reviewable service definition

**Compose files**

authored intent

→

**config**

resolved model

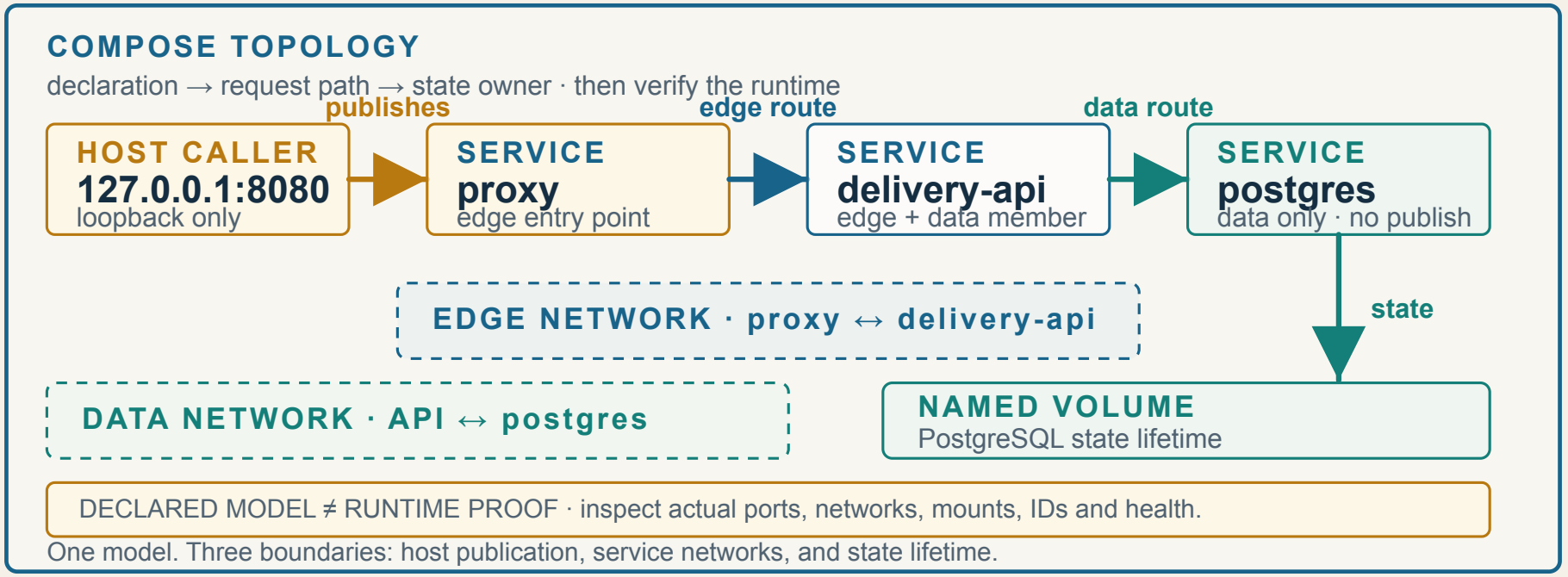
→

**review gate**

ports · networks · volumes · probes

Review the effective Compose model before changing runtime state.

# Declared topology versus runtime evidence




The model declares proxy → API → PostgreSQL, two networks, one named volume and one host publication; runtime checks verify what was created.

# A service is a role; a container is an instance

SERVICE ROLE · DESIRED

## delivery-api

image coordinate · networks · mounts · health check · policy

recreate →

INSTANCE · OBSERVED

## container B

ID · process state · health history · timestamps · actual attachments

Recreating an instance does not erase the service role or prove that its attachments stayed correct.

# Running, healthy, ready and CRUD are four claims

PROCESS

## running

PID exists; no request guarantee

LOCAL HEALTH

## /healthz · 200

handler responds locally

DEPENDENCY READINESS

## /readyz · 503

dependency-backed check fails

BEHAVIOR

## CRUD test

schema and transaction path

A green local health endpoint cannot certify readiness or successful CRUD.

# Readiness is a request contract with limits

READINESS

## /readyz → 503

dependency cannot satisfy SELECT 1 now

REGRESSION

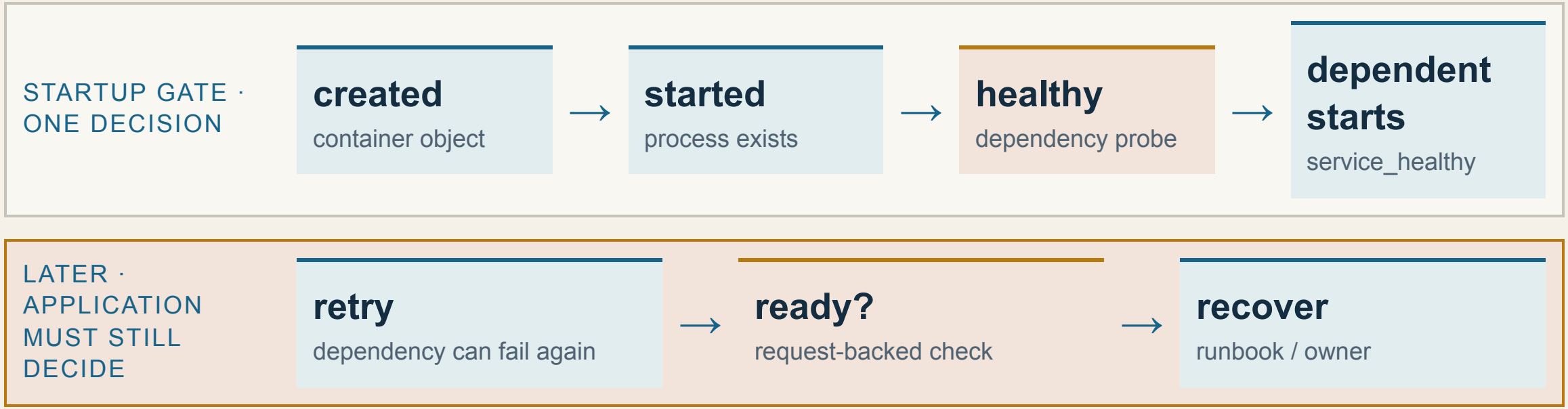
## CRUD → ?

schema, write, read and cleanup still need evidence

A readiness result is bounded by the exact dependency/query it exercises.

Author & Course Lead — Badr Tajini · ESILV · 2026–2027

# A healthy dependency gates startup once




service_healthy reduces a startup race; it is not a continuous repair controller.

# Startup readiness and later retry are different controls

**Startup gate**

wait for dependency health

→

**Application**

retry / report readiness

→

**Operator**

capture and decide

A health-aware start order does not replace application retry or an operator-owned recovery path.

# A probe must name the condition it measures

A USEFUL LOCAL CONDITION

**Process**

handler accepts the probe

**Dependency**

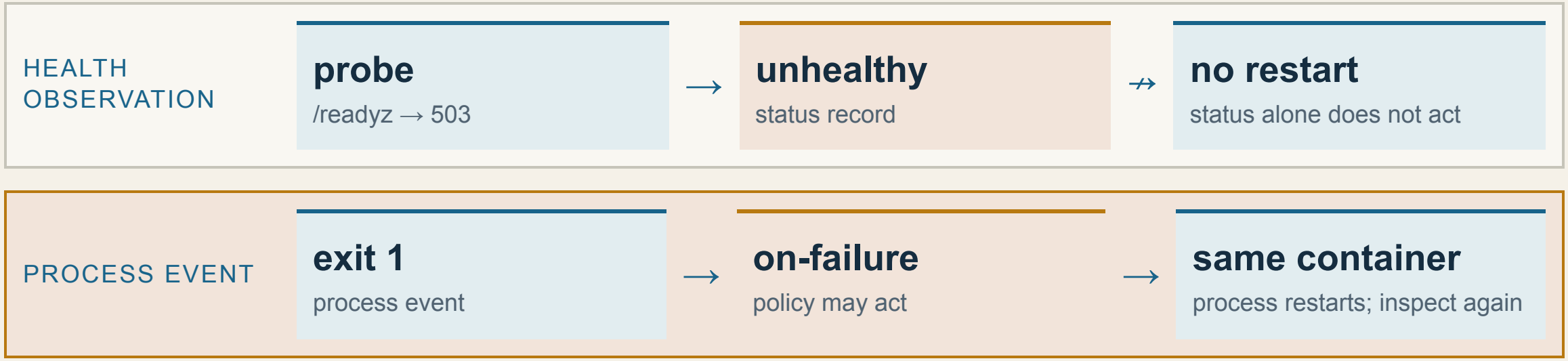
declared query succeeds

**Limit**

CRUD and every request remain untested

A health check is valuable when its condition, dependency and limitation are written down.

# Unhealthy alone does not trigger a process restart




Health status and process-exit policy are different mechanisms.

FAILURE CLASS

# A restart policy cannot repair every failure

MAY RECOVER

## process crash

policy can restart the same container

NEEDS DIAGNOSIS

## bad config

password · schema · disk · route

Automatic process recovery is not configuration or data repair.

Author & Course Lead — Badr Tajini · ESILV · 2026–2027

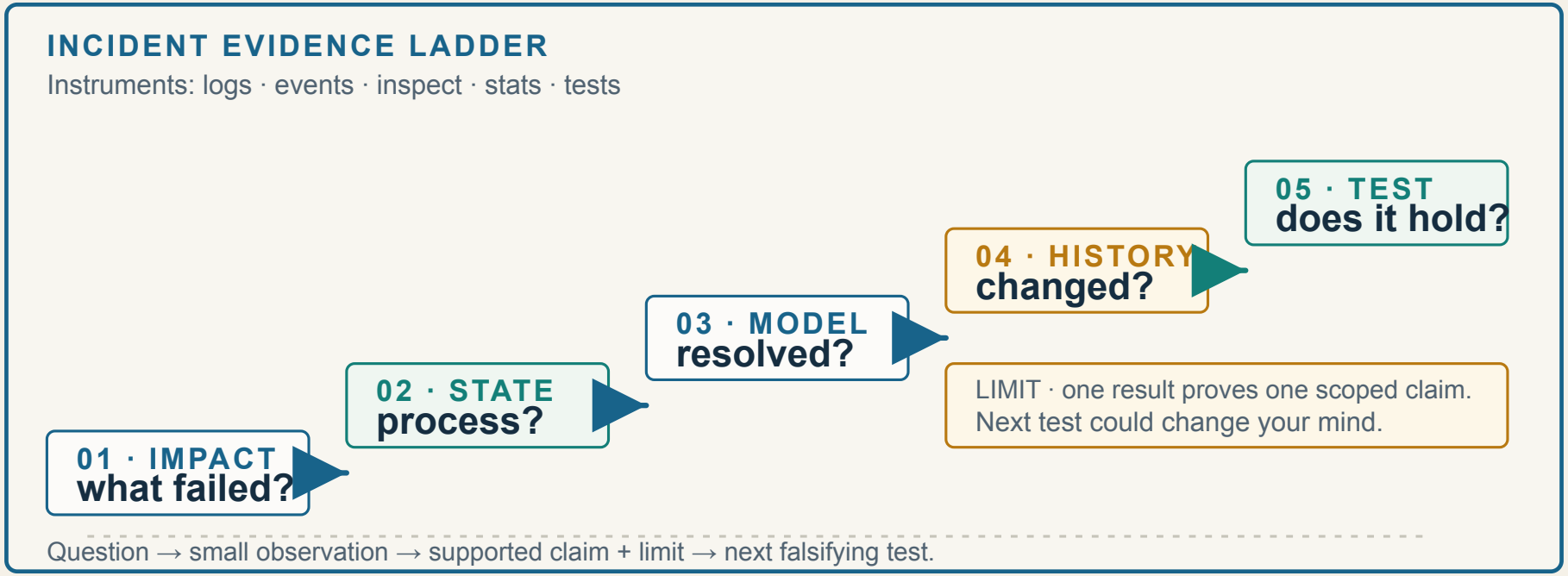
# Preserve the first failure before mutation

REFERENCE TRANSCRIPT · not observed execution

```
$ docker compose -p docker-delivery ps --all
$ docker compose -p docker-delivery logs --since 5m --no-color
$ docker inspect --format '{{.State.ExitCode}} {{.State.Health.Status}}'
delivery-api
REFERENCE ONLY · command shapes, no observed provider result
```

Capture bounded state before a restart so the event, health result and definition can be compared.

# Four views, five instruments, one claim




A useful evidence set connects impact, runtime state, definition and change history.

# Evidence is a chain, not a pile

**Question**
what could be wrong?

→

**Observation**
small bounded result

→

**Interpretation**
supported + limited

→

**Next test**
falsify or refine

Interpretation makes an observation useful; the limitation keeps it honest.

# Ten minutes to reset

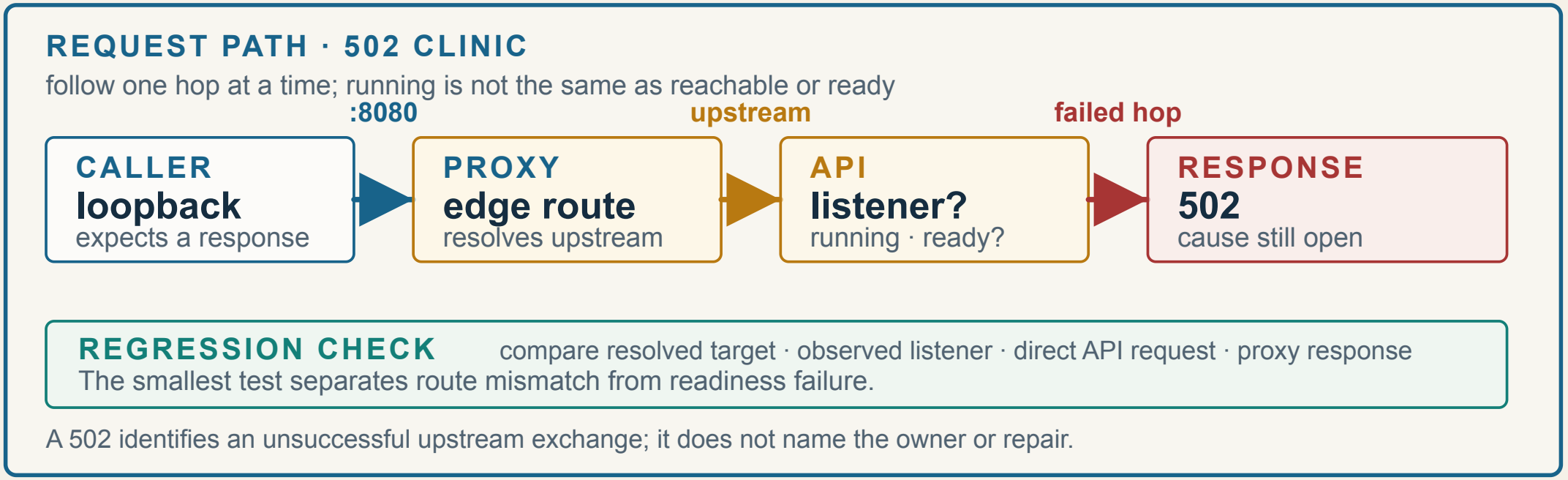
## Return with one question: which evidence view would change your mind?

Return in ten minutes.

A real break protects attention before the proxy clinic and diagnosis experiment continue.

Author & Course Lead — Badr Tajini · ESILV · 2026–2027

# A 502 is a failed hop, not a diagnosis




Follow the request path one hop at a time before changing the stack.

# Compare the declared target with the runtime route

DECLARED

## proxy → delivery-api

edge name + container port

OBSERVED

## upstream exchange

proxy log + API state + exact time

A route diagnosis compares the effective model with scoped runtime evidence.

# A failure experiment changes one variable

**Baseline**
positive + negative checks

→

**Fault**
one scoped definition

→

**Observation**
bounded window

→

**Restore**
known state

A useful experiment has a baseline, one deliberate change, a short observation window and a restoration step.

# Observe before you mutate

**ILLUSTRATIVE COMPLETED CASE · proxy 502**
**OBSERVE:** proxy returns 502; API process is running.
**TEST:** resolved upstream is delivery-api:8001; API listens on 8000 and its direct request returns 200.
**OWNER:** versioned proxy route definition.
**REPAIR:** correct the upstream port to 8000.
**REGRESSION:** proxy request now returns 200; repeat health, readiness and CRUD checks before closing.
This is a worked example, not measured runtime evidence.

Test one explanation before changing one owned definition.

# Hypothesis → test → small repair → regression

**Hypothesis**

wrong upstream coordinate

→

**Test**

compare model + route

→

**Repair**

change one owner

→

**Regression**

repeat 200 / 503 / CRUD claim

A repair follows evidence about the fault and is checked by the original regression.

# An incident record is a replayable decision

| IMPACT / TIME | SCOPE | EVIDENCE | DECISION | RECOVERY |
|---|---|---|---|---|
| **what failed · UTC start** | **project · service · window** | **ps · logs · model · events** | **hypotheses · test · owner** | **one repair · regression · risk** |

Record impact, scope, evidence, hypotheses, owner, repair, regression and residual risk.

Author & Course Lead — Badr Tajini · ESILV · 2026–2027

# Ownership makes cleanup safe

CAPTURE FIRST

## ps · logs · model · events

preserve state and timing

CHANGE ONE

## owner · version · scope

never global prune or down --volumes

Scoped recovery requires evidence, ownership and an explicit rollback or cleanup limit.

# A rollback needs immutable coordinates


A known-good full Git SHA and API image digest make a rollback target inspectable.

# Rollback is a replayable definition change

**Record**

current + known-good

→

**Worktree**

detached target

→

**Resolve**

effective model

→

**Verify**

same endpoint + CRUD

A rollback is an executed, evidence-captured procedure with a declared final state.

# Do not repair before a falsifiable test

## A service is unhealthy. What comes before restart or recreation?

> State the failed expectation, bound the component and time, capture evidence, rank hypotheses and run the smallest test that could change your mind.

A repair is safer after a bounded test identifies a likely owner and a residual risk.

# Small repair, same claim, explicit limit

**Evidence**
owner identified

→

**Repair**
one versioned change

→

**Regression**
same positive + negative

→

**Limit**
what remains unknown

The regression must test the original user expectation and name what it does not cover.

# TD4 starts with a reviewable Compose model

**CLAIM**
expected behavior

→

**CAPTURE**
bounded record

→

**REASON**
hypothesis + owner

→

**REPAIR**
one versioned change

→

**REPLAY**
same checks + limit

TD4 carries the model, evidence, reasoning, repair and replay into an assessed failure clinic.

Author & Course Lead — Badr Tajini · ESILV · 2026–2027

# Choose the fact that makes repair safer

## Which fact makes the next repair safer?
## failing command · first log line · resolved model · owner · known-good identity

> The answer depends on the hypothesis, but the safest choice is the fact that bounds the owner and lets the same claim be replayed. Explain what it proves and what remains unknown.

---

A good operator names the evidence that constrains the next action and its limitation.

# Leave a replayable state for TD4

**Submitted state**
definition + identity

→

**Evidence**
checks + record

→

**Residual risk**
named limitation

→

**TD4**
guarded clinic

A clean handoff states what was submitted, what was tested and what remains uncertain.

# Contingency and remote recovery

USE IF NEEDED

## replay one visual

answer a route question or recover a remote pause

CLOSE

## declare state

retain evidence and transition to TD4

Ten minutes remain available for a meaningful replay, accessibility pause or route recovery.