The image stops before runtime

Image

packaged content

→

Runtime

process + choices

→

Evidence

what was tested

An image is an input to a runtime contract, not the whole deployment.

What must survive replacement?

A container is replaced from the same image.
Which object should keep a delivery record?

> The named data volume or another explicitly managed state store. The container writable layer belongs to the old instance.

Survival is a property of an explicitly chosen state owner.

Follow the request and the data separately

REQUEST PATH

Browser → API

a reachability question

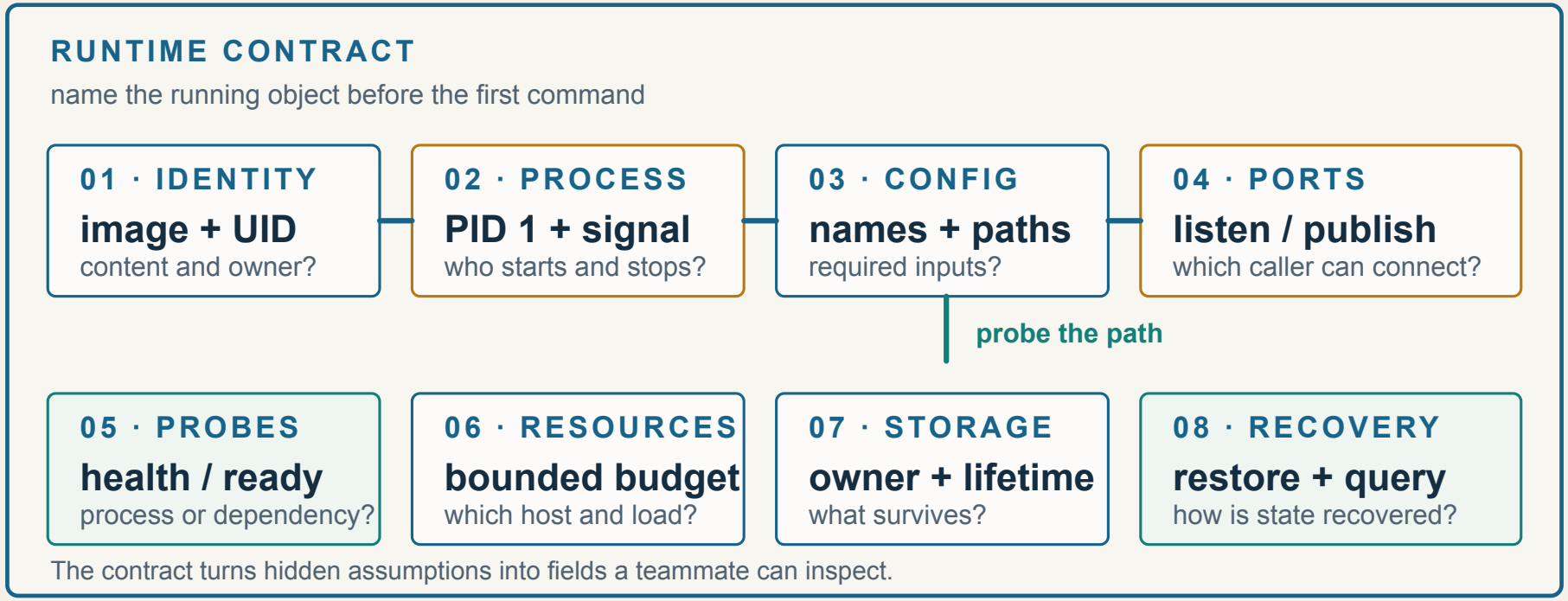
DATA PATH

API → PostgreSQL

a persistence question

A network path makes a service reachable; a volume gives state a lifecycle.

Eight questions before start



Runtime contract: identity and process; configuration and ports; probes and resources; storage owner and recovery query. These eight fields make startup assumptions reviewable.

A runtime contract turns hidden assumptions into reviewable fields.

Identity, configuration, publication

Digest
which content?

→

UID / PID 1
which process?

→

Variables
which inputs?

→

Host binding
which audience?

A port number without caller, interface, and process context is incomplete.

Health is not readiness

PROCESS HEALTH

/healthz → 200

the application process responds

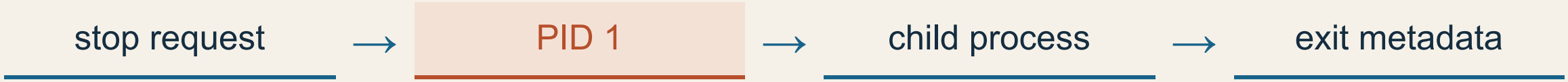
DEPENDENCY READINESS

/readyz → 503

PostgreSQL is unavailable

A running process can be healthy while the dependency-backed request is not ready.

SIGTERM has a destination



The configured grace period bounds the transition.

A stop request is meaningful only when PID 1 handles the configured signal.

A stop record is evidence, not a story

REFERENCE TRANSCRIPT · not observed execution

```
reference $ docker inspect delivery-api
configured signal: SIGTERM
exit metadata: recorded
stop budget: 15s
REFERENCE ONLY · no Docker daemon was invoked
```

A transcript becomes useful when its identity, expected result, and limits are explicit.

Shutdown is not persistence

The process exits cleanly.
Did the record survive?

> Unknown until the state owner is read after replacement or restore. A clean stop is a process claim, not a storage claim.

Process lifecycle and state lifecycle require different evidence.

Three kinds of runtime input

IMAGE DEFAULTS

baked

part of packaged content

RUNTIME VALUES

named

configuration and secret path

Configuration is supplied at runtime; secrets cross a narrower, named boundary.

Name the path, never the value

Runtime definition

POSTGRES_PASSWORD_FILE

→

Mounted path

/run/secrets/db_password

→

Authorized process

PostgreSQL only

→

Evidence

names + permissions

A secret contract records the access path and authorized consumer, not the secret itself.

A good transcript omits the secret

REFERENCE TRANSCRIPT · not observed execution

```
reference $ inspect runtime definition
DATABASE_HOST=postgres
DATABASE_PORT=5432
POSTGRES_PASSWORD_FILE=/run/secrets/db_password
password value: [omitted by contract]
```

Evidence can prove names, paths, and access boundaries while withholding secret values.

Whose localhost?

HOST

`127.0.0.1`

host loopback

API CONTAINER

`127.0.0.1`

API loopback

`localhost` belongs to the current caller’s network namespace.

Names survive replacement better than IPs

API
caller

→

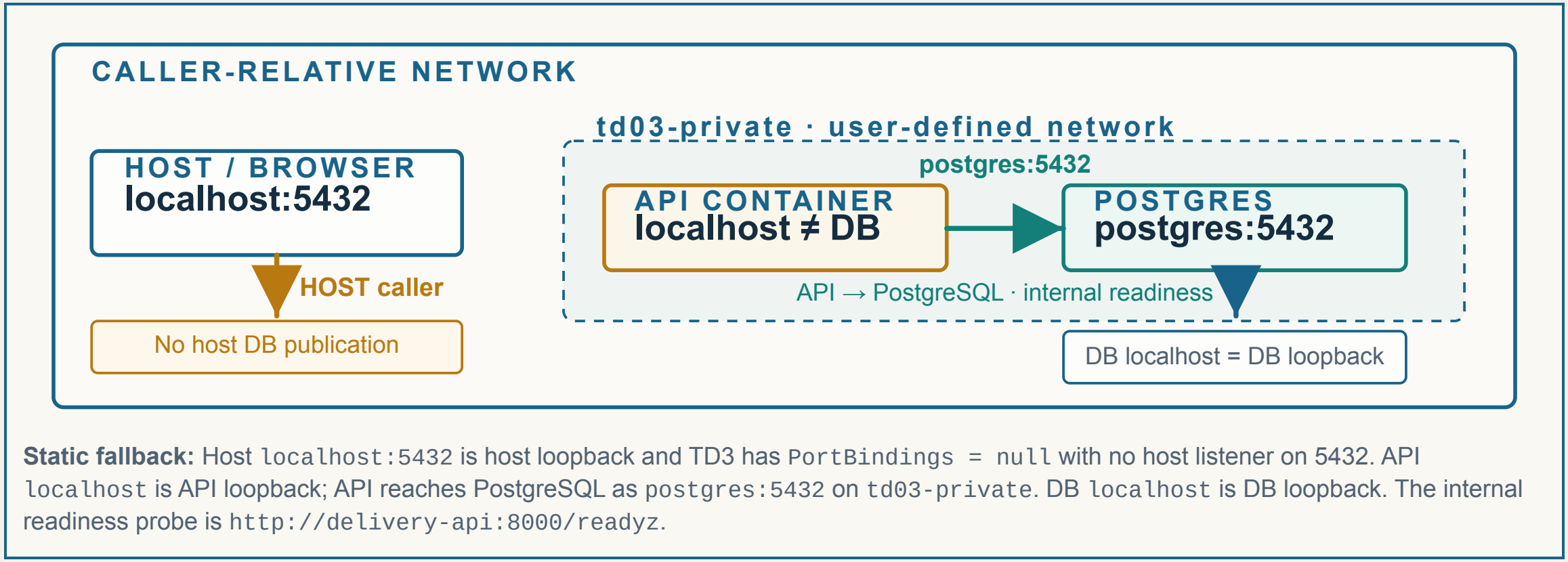
td03-private
user-defined network

→

postgres:5432
service name + port

TD3 uses `postgres:5432` on `td03-private`; a recreated IP is an observation, not the contract.

Change the caller, change the meaning



Static fallback: Host `localhost:5432` is host loopback and TD3 has `PortBindings = null` with no host listener on 5432. API `localhost` is API loopback; API reaches PostgreSQL as `postgres:5432` on `td03-private`. DB `localhost` is DB loopback. The internal readiness probe is `http://delivery-api:8000/readyz`.

The target is caller-relative: API uses postgres:5432 on td03-private for PostgreSQL.

Reachable is not ready

The API can resolve `postgres`. Is `/readyz` guaranteed to return 200?

> No. DNS reachability is one condition; PostgreSQL must also accept the dependency-backed query now.

Name resolution proves a path to a peer, not successful application behavior.

A port has three meanings

Process	Network	Publication
where it listens	which peer can connect	which host path exists

`5432` can be listening internally without being published on the host.

API reaches PostgreSQL without a host port

TD3 keeps API and PostgreSQL on one private network and publishes neither service.

Prove the path and the absence

POSITIVE

API → postgres:5432

DNS + dependency query

NEGATIVE

PortBindings = null

no host listener in TD3 check

A bounded nonpublication claim needs an internal success and an explicit absence check.

Ten minutes to reset

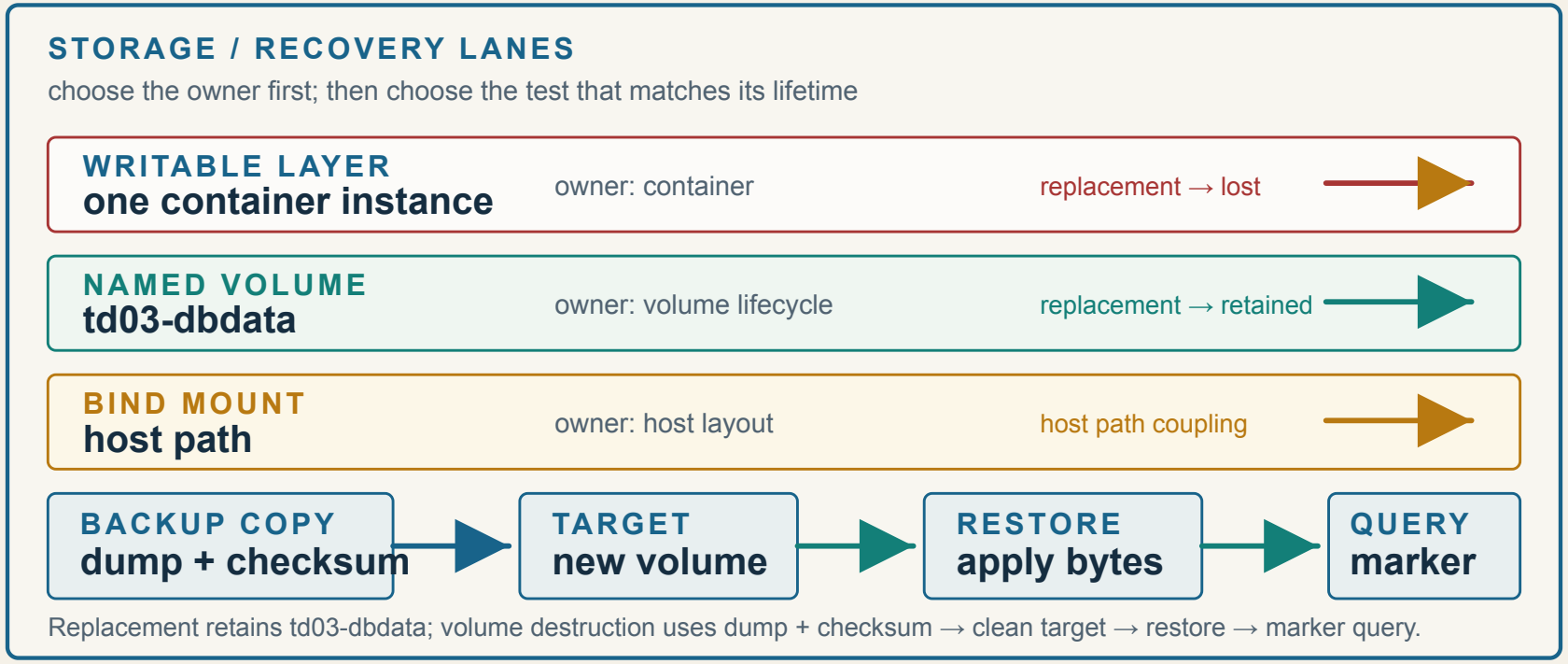
Return with one question: whose namespace owns this address?

Return in ten minutes.

A real break protects attention before storage, persistence, and recovery continue.

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

Four storage lanes



Static: replacement retains `td03-dbdata`; volume destruction needs dump + checksum → clean target → restore → marker query.

The storage label is incomplete until its owner and lifecycle are named.

State belongs somewhere specific

Container layer
old instance owns it

→

td03-dbdata
volume owns it

→

Host path
bind mount owns it

→

Dump + checksum
recovery owns it

A replacement test only supports the state claim for the owner you actually attached.

Where should this file live?

A cache, a database row, and a restore dump are written. Choose a lane for each.

> Cache → writable layer; database row → named volume; restore dump → separate recovery artifact. A host bind mount is a deliberate host-coupling choice, not the default answer.

Storage choice follows the required lifetime and recovery action.

A persistence test has a sequence

Write
known marker

→

Record
before bytes

→

Replace
container only

→

Read
after bytes

→

Compare
exact equality

Persistence is supported by a before/after comparison across container replacement.

Same marker, different container

REFERENCE TRANSCRIPT · not observed execution

```
reference $ compare marker records
before: 1|TD03-PERSIST
after:  1|TD03-PERSIST
result: exact bytes equal
REFERENCE ONLY · supplied sequence, not observed here
```

Marker equality supports the tested volume-persistence claim and nothing wider.

What does the marker prove?

The marker survives replacement.
Which claim is justified?

> The tested named volume retained that marker while the container was replaced. It does not prove a full backup, a clean restore, or every database record.

A small test can support a precise claim without becoming a system guarantee.

A checksum is an identity check

COPIED BYTES

`delivery.dump`

nonempty + contents list

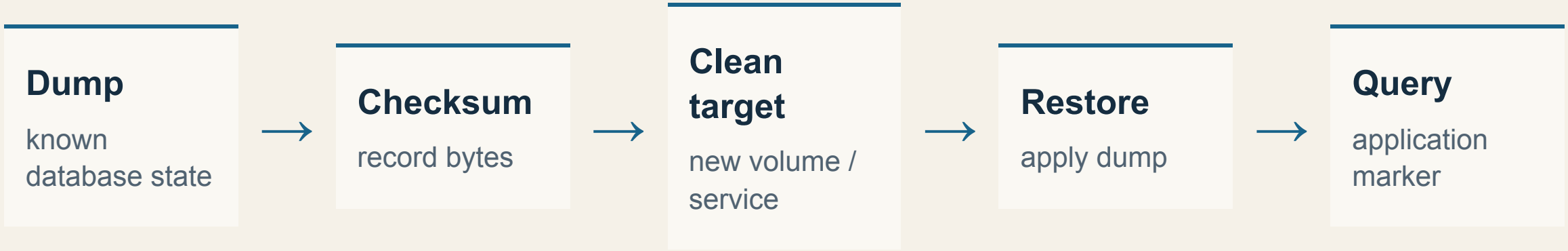
RECORDED IDENTITY

`sha256`

bytes match the record

A checksum verifies the dump bytes against a recorded checksum; it does not verify usable application state.

Restore into a clean target



Recovery is established by a successful restore followed by an application-level query.

Recovery ends with a query

REFERENCE TRANSCRIPT · not observed execution

```
reference $ recovery evidence
dump: nonempty
checksum: SHA-256 match
restore target: clean
marker-after-restore: 1|TD03-PERSIST
REFERENCE ONLY · no volume was deleted here
```

The restore query supports usable state; the reference transcript itself is not provider evidence.

A limit is an assumption to measure

CPU	Memory	PIDs	Observation
share / quota	startup + request	process budget	host + workload

A configured limit constrains consumption; it does not guarantee workload success.

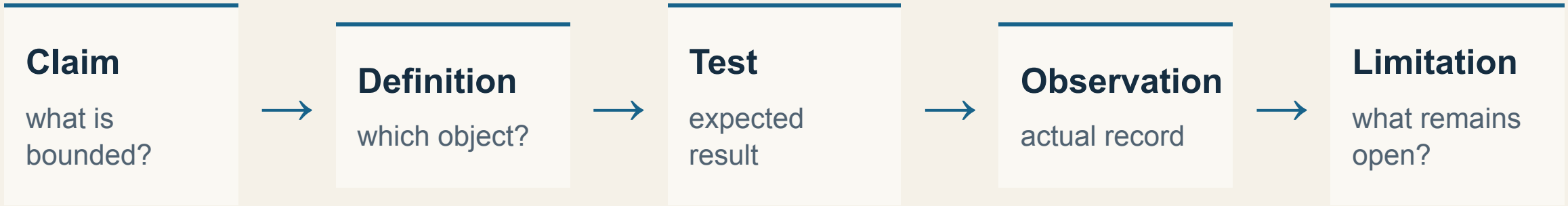
What did the limit establish?

The service stayed under a 256 MiB limit once. What can you claim?

> The recorded workload and host stayed under that configured limit during the observation. You cannot generalize to every load or prove the limit caused another outcome.

Measurements support bounded observations, not universal performance promises.

A claim needs a replayable path



A screenshot is supporting evidence only when its identity, expectation, result, and limitation are known.

Replace ‘it is secure’ with a testable sentence

The container runs, so ‘it is secure’. Repair the claim.

> Example: ‘In the TD3 check, the API reached PostgreSQL on `td03-private`, and PostgreSQL had no host PortBindings; host-route behavior outside that check remains outside the claim.’

A narrower claim is stronger because its evidence can actually refute it.

TD3 starts with four contracts

Private path	Secret boundary	State owner	Recovery proof
API + PostgreSQL	path, no value	td03-dbdata	checksum + query

TD3 turns these four models into exact definitions, tests, and retained evidence.

Leave with a question you can test

Predict
caller + object

→

Run TD3
exact definition

→

Record
result + limit

→

Recover
repeat the contract

CM3 supplies the model; TD3 supplies the controlled runtime and evidence pack.

Use the buffer to repair one boundary

Context · object · evidence

Bring one unresolved question back to the group.

The buffer absorbs a slow prediction, a reference-transcript explanation, or a short retrieval exchange.