A release needs a way back

Change
source revision

→

Candidate
tested identity

→

Recovery
previous known-good

A delivery decision is incomplete until its release identity and recovery path are explicit.

CI feedback is not approval

Tests pass on a change.
What decision is still outside CI?

> A human still decides whether the candidate may be promoted to the target. Publication makes an artifact available; it does not grant deployment approval.

A successful check supports a claim; it does not silently grant permission to deploy.

Name the trigger before the incident

PROMOTE

smoke passes

candidate remains active

STOP / RETURN

explicit trigger

retain P and reverse safely

A rollback is a response to a named trigger, not a vague feeling that the release looks wrong.

Integrate small changes into a usable mainline

Coursework branch
shared integration point

→

Automated feedback
tests + policy

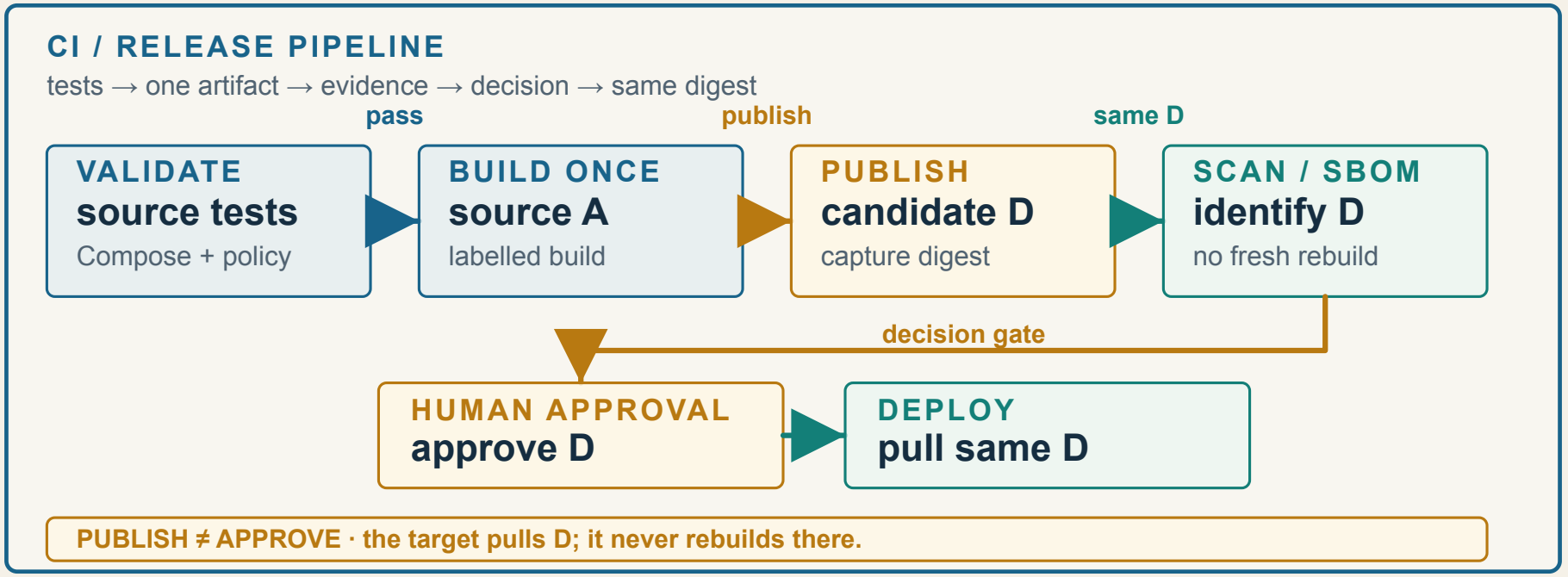
→

Prompt repair
keep system usable

CI is a shared working practice with automated feedback, not a long-lived private branch with tests attached.

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

Tests come before the candidate leaves CI



The candidate is built once only after source and definition checks pass, then the same digest flows through scan and SBOM.

Which event may publish?

A pull request changes application code. Which action should it receive?

> Validation and policy checks. The pull request must not publish a registry candidate or receive deployment credentials.

Trigger and permission boundaries make the CI claim testable.

Four jobs, four kinds of evidence

validate
tests + Compose + policy

→

build-publish
one build + digest

→

supply-chain
scan/SBOM D

→

release-candidate
summary; no deploy

A pipeline job is useful when its output and gate are explicit.

Label the image with the source that built it

SOURCE A

full 40-char SHA

workflow head + OCI revision

IMAGE D

digest

immutable content identity

The image carries the source revision it was built from; the digest identifies the resulting bytes.

Publication is a candidate, not a decision

PUBLISHED

candidate D exists

available for inspection

≠

APPROVED

human accepts D

deployment may begin

→

DEPLOYED

same D on target

smoke still required

A published candidate is inspectable; a human approval opens the deployment path.

Untrusted code does not receive deploy authority

UNTRUSTED PR

tests + static review

no registry write · no deploy credential

→

PUBLISH JOB

GITHUB_TOKEN

packages: write only here

→

HUMAN GATE

approve digest D

deployment credential remains withheld

Pull requests validate with no registry write or deployment credential; publish and deploy permissions stay behind separate gates.

Grant the smallest permission at the smallest job

DEFAULT

contents: read

workflow starts narrow

PUBLISH JOB

packages: write

only where image publication occurs

A permission review asks where authority begins, why it is needed, and where it is absent.

No secret crosses into an untrusted build

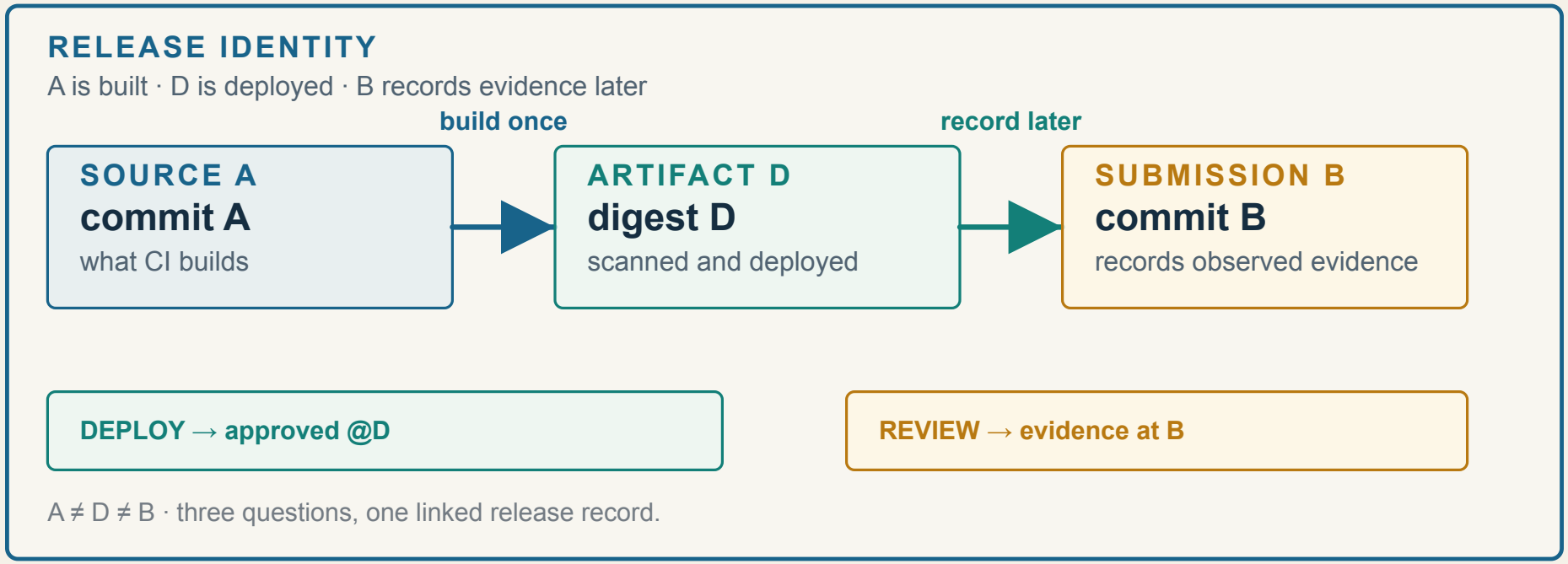
An untrusted pull request changes the Dockerfile. What must the build not receive?

> Registry write credentials, deployment credentials, SSH keys, private environment values, and any host-authority socket. The build gets only the intended context and validation permissions.

Build isolation is a boundary around inputs and authority, not a promise that a Dockerfile is harmless.

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

A, D, and B answer different questions



Source commit A, image digest D, and later submission commit B are distinct identities in one release record.

One build, one digest, many later consumers

The same published digest travels through scan, approval, deployment and rollback evidence.

/version is a link, not independent byte proof

VERSION

release_sha = A

application-reported metadata

PROOF CHAIN

workflow → registry → inspect

endpoint confirms behavior and metadata only

A correct-looking version response can be copied or configured; link it to workflow, registry, and container evidence.

Approval is a written engineering decision

APPROVAL RECORD

Candidate
source A + digest D

Gates
tests + scan/SBOM references

Target
environment + timestamp

Recovery
previous P + trigger

Approval joins evidence, target, approver, timestamp and rollback trigger.

Would you approve this candidate?

Candidate D was published and scanned.
The manifest names D, but the approval omits the previous digest and rollback trigger.
What is missing?

> The release decision lacks a known recovery target and an explicit stop condition.
> Publication and scan do not replace that decision.

A candidate can be technically valid and still fail the operational approval gate.

Midpoint break — return with the approval question

MIDPOINT BREAK

10 minutes

Return ready to trace approved digest D into deployment.

Take ten minutes, then return ready to trace approved digest D through deployment and rollback.

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

Deploy the approved digest, never a moving tag

REGISTRY

ghcr.io/course/api@sha256:D

pull exact approved digest

→

RUNTIME

Config.Image → … @D

approved digest-qualified reference

→

SMOKE

/healthz + /version

behavior evidence, not byte proof alone

The target pulls D; inspect Config.Image shows the approved reference, and smoke tests the runtime contract.

A green build does not prove a healthy target

CI

tests + scan + SBOM

candidate evidence for D

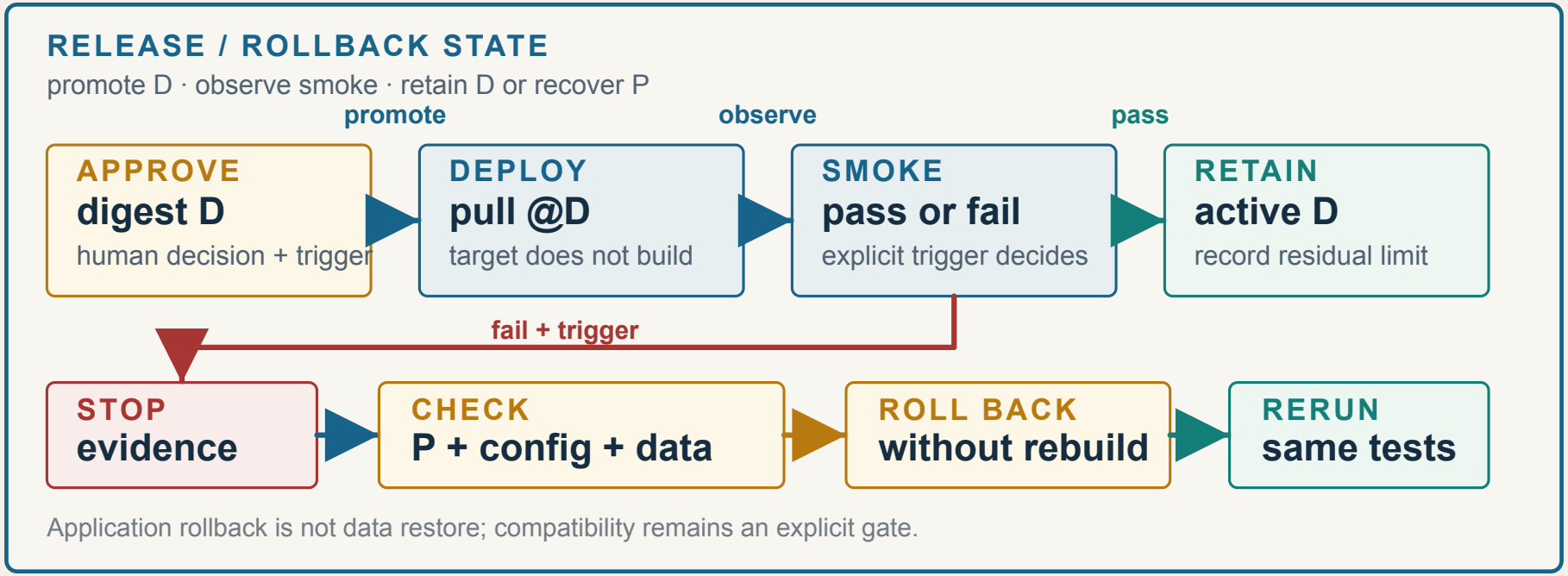
TARGET

smoke + regression

runtime behavior for D

CI and target smoke answer different questions and both belong in the release record.

Rollback needs more than an older tag



Retain previous digest P, its definition, and data-compatibility decision before promotion.

Manually walk one release and one recovery

Static trace: source tests pass for A → build once → publish and scan/SBOM D → human approve D with P retained and definition/data compatibility checked → deploy D → smoke passes. If the explicit smoke trigger fires, recover P with compatible definition/data, without rebuilding, then rerun the same checks.

Approve and deploy the same digest; recover only through a compatible retained release.

Application rollback is not data restore

The old image is compatible with the database contract, but a migration is not backward-compatible.
Can application rollback alone recover the service?

> No. The runbook must stop or use the documented data recovery path. Application identity and database state are related but distinct recovery claims.

Rollback of an image does not silently restore data or undo an incompatible migration.

Link workflow, registry, container, endpoint

A release claim is stronger when independent systems link source A, digest D, active container and behavior.

The endpoint cannot certify its own bytes

SUPPORTED

/version → A

application reports a revision

STILL NEEDED

registry + inspect → D

independent content identity

The version endpoint supports metadata and behavior claims; registry and container identity close the byte-link claim.

Kubernetes consumes the contract later

DOCKER CONTRACT

digest · UID · ports · probes · resources · config · data · shutdown

→

LATER KUBERNETES CONCERN

image · security context · Service · probes · requests/limits · ConfigMap/Secret · PVC · lifecycle

CM6 hands over a runtime contract; it does not teach Kubernetes administration or manifests.

Do not smuggle assumptions into Kubernetes

HAND OVER

identity · probes · resources · state

contract fields with evidence

STOP HERE

no manifests · no HA claim

later course owns orchestration

A precise handoff states what is known and what the Docker single-host route cannot guarantee.

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

Make one final release decision

Candidate
A · D · gates

→

Human
approve / reject

→

Target
same D + smoke

→

Recovery
P + trigger

Release approval is a decision supported by linked evidence and a rehearsed recovery path.

Find the broken link

Workflow source is A; scan names D; target inspect shows D; approval names a different digest; /version reports A. What is the safe decision?

> Stop. Approval and target identity do not agree with the candidate record. Do not treat the matching version response as proof; reconcile the approval, digest, and deployment record or roll back.

A single inconsistent identity blocks promotion even when several other checks look green.

The Docker Delivery Pack is a claims index

PLAN
invariants + runbook

→

CI
run + digest + SBOM

→

APPROVAL
human decision

→

RECOVERY
rollback + checksums

TD6 packages plan, CI evidence, approval, runtime proof, recovery, checksums and the runtime contract.

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

Explain the failed expectation and residual risk

In the individual defense, what must you explain after rollback?

> The failed expectation, the inspection and hypothesis, the repair or rollback, the same acceptance test, and the residual risk/evidence produced.

A release defense is a claim-and-evidence explanation, not a command recital.

Leave a replayable release state

Submitted
commit B + pack

→

Deployed
digest D or P

→

Evidence
checksums + links

→

Limit
named residual

A clean close states which commit, digest, evidence and limitation are being handed to TD6.

Contingency and remote recovery

USE IF NEEDED

replay one gate

identity, approval, smoke, or rollback question

CLOSE

declare state

retain evidence and transition to TD6

Ten minutes remain for a meaningful replay, accessibility pause, CI/registry recovery discussion, or final-state declaration.