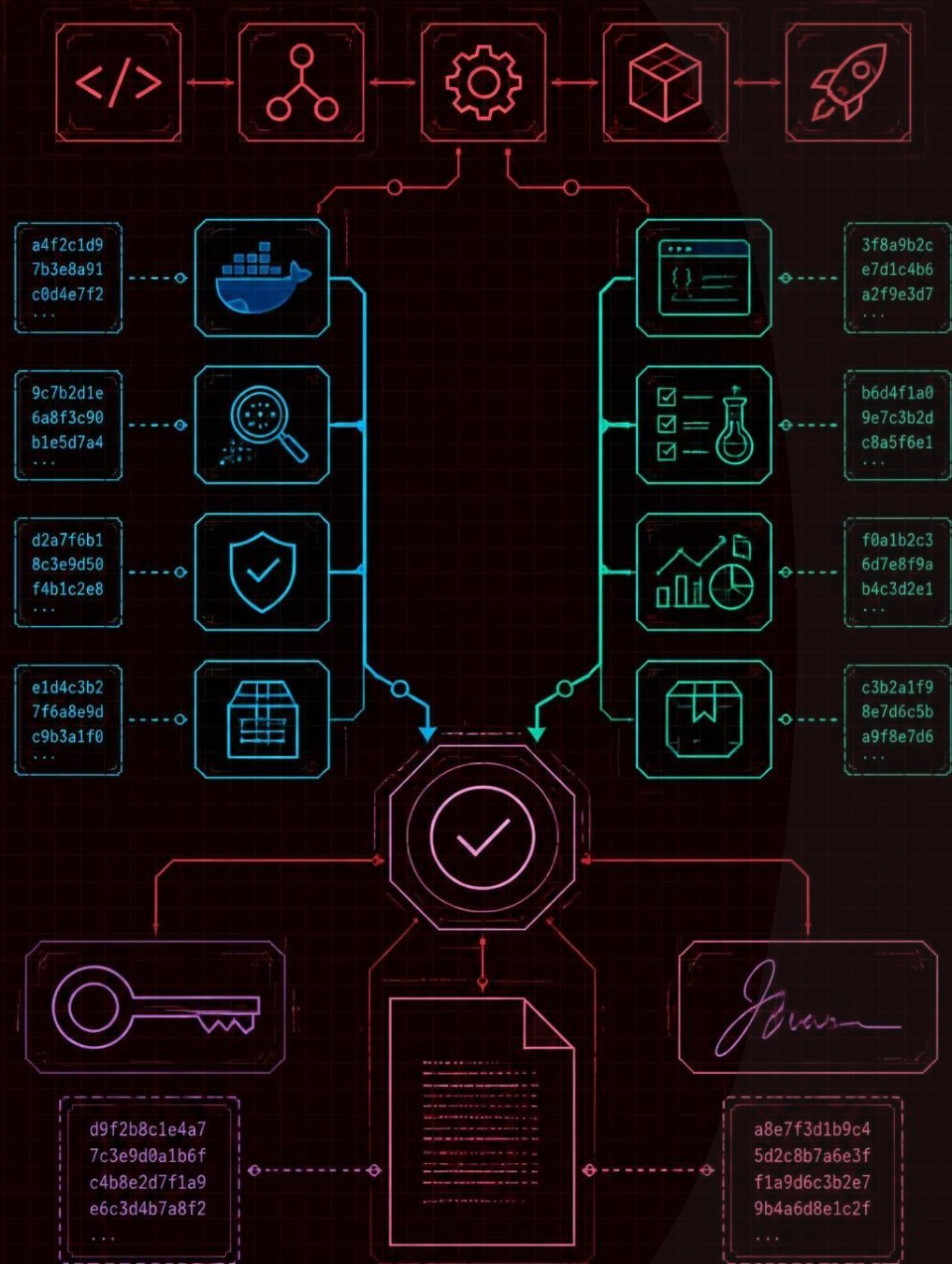




FACTI · BRAZIL | ENGINEERING EXPERIENCE REPORT

From Signature Generation to Operational Verification

Deployment Lessons from Hybrid Post-Quantum Software Signing in CI/CD

Rafael Lucas Silva, Luiz Otavio Duarte, Larissa de Oliveira Figueira. Facti, Brazil

Signing Is Only the Beginning

Signatures establish authenticity and integrity of packages, container images, firmware, and manifests.

01

Build

Create the artifact.

02

Sign

Generate evidence.

03

Distribute

Deliver both.

04

Verify

Apply trust policy.

05

Archive

Preserve evidence.

Verification may happen offline or many years later.

How can post-quantum signatures become operationally trustworthy across heterogeneous software supply chains?

Hybrid Signing in GitLab CI/CD: Two Independent Signatures

Build + manifest →

GitLab CI/CD builds the artifact

Deterministic manifest of SHA-256 file hashes; compute the manifest digest.

Two independent signatures →

Both sign the same digest independently.

Classical signature

HashiCorp Vault Transit

Post-quantum signature

Dilithium-family via Open Quantum Safe (liboqs)

One signature bundle

Both signatures + metadata go into one bundle.

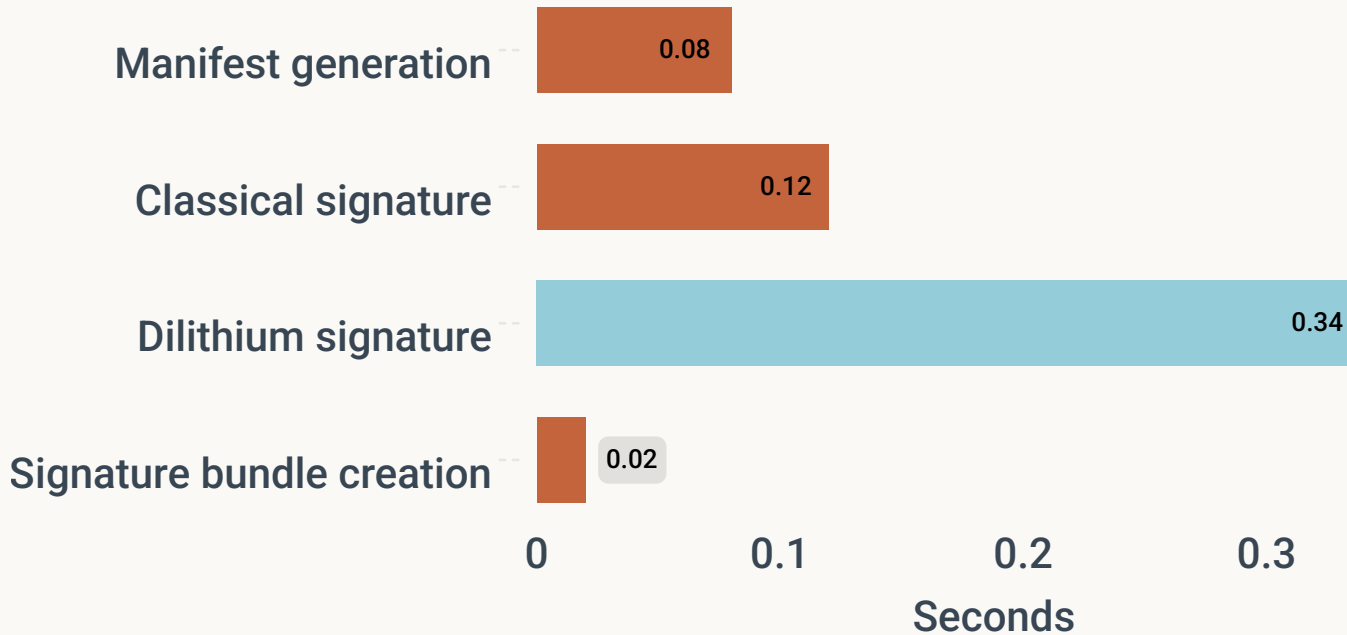
Consumers apply their verification policy. Classical consumers keep working; PQ-capable consumers can also verify the PQ signature.

Hybrid at the verification-policy layer, not a composite signature. Experimental prototype, not standardized ML-DSA and not production-ready.

Performance Was Not the Main Bottleneck

GitLab Runner · 4 vCPUs · 8 GB RAM · Ubuntu 24.04 · artifact of about 12 MB

Operation



0.56 s

Total additional time

Small against CI/CD runs measured in minutes.

Specific to the evaluated configuration, not a universal guarantee.

Signing overhead per operation (seconds). Dilithium signing was the largest single step.

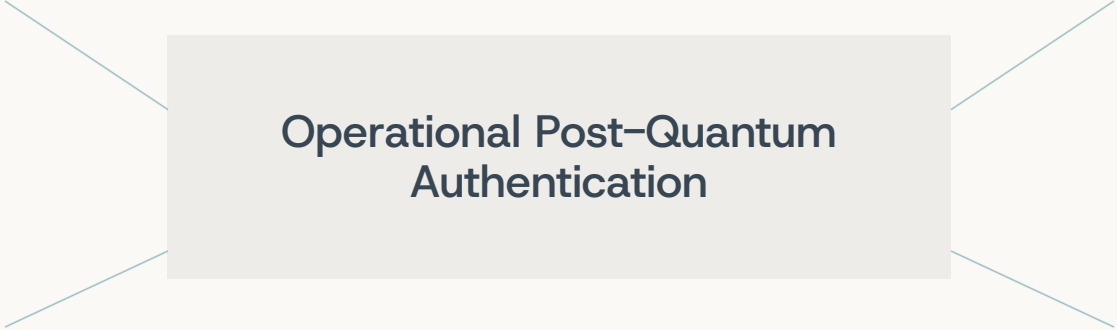
The Real Challenges Begin After Signing

Verification semantics

Must both validate? Can one suffice? What if PQ support is absent?

Key management

Vault supports classical custody; PQ needs an equivalent operational lifecycle.



The diagram features a central light gray rectangular box with the text "Operational Post-Quantum Authentication". Four thin blue lines radiate from the corners of this box to the corners of a larger, faint gray rectangle that frames the entire content area, connecting the central concept to the four surrounding challenge sections.

Operational Post-Quantum Authentication

Consumer migration

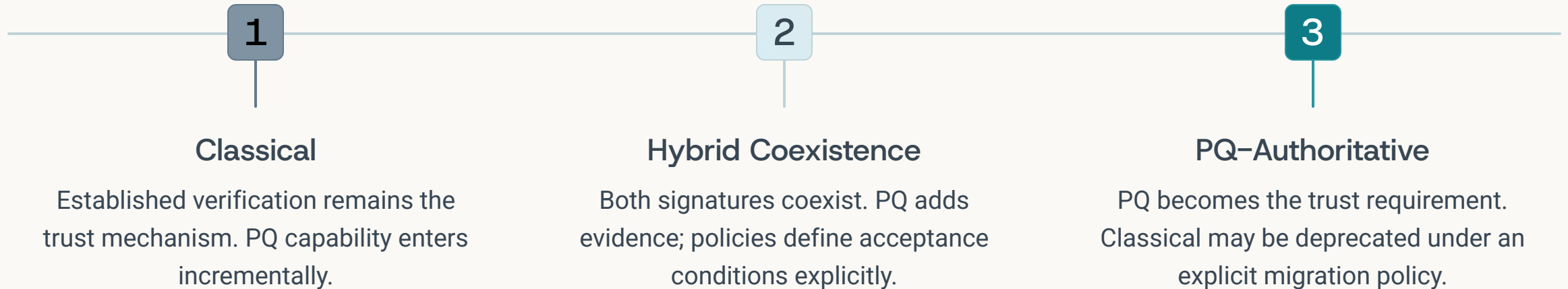
Registries, deployment tools, package managers, and validators update at different speeds.

Long-term verification

Evidence must survive key rotation, certificate expiry, and algorithm transitions.

The complexity moves from cryptographic implementation to operational trust management.

Hybrid Signing Enables Incremental Migration



Hybrid deployment creates migration flexibility, but interoperability depends on explicit verification policies.

Conceptual migration stages, not an implemented or experimentally validated framework.

What Is Still Missing for Operational PQ Authentication?

Hybrid signing profiles

How should classical and PQ signatures bind unambiguously to the same artifact?

Verification policy evolution

How can acceptance rules evolve from classical-only through hybrid to PQ-authoritative?

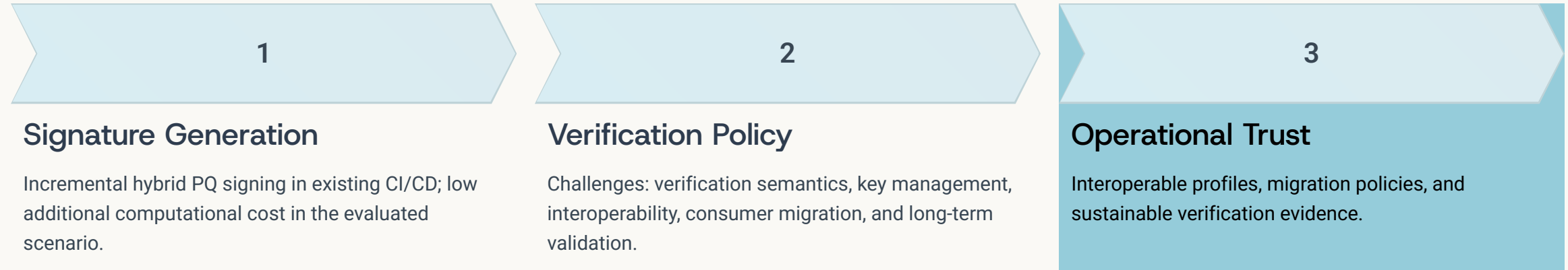
Long-term authentication evidence

Which metadata and evidence must survive for offline and archival validation?

Supply-chain integration

How should signing fit SBOMs, provenance attestations, transparency systems, secure update frameworks, and package and container signing?

From Post-Quantum Signatures to Post-Quantum Trust



Generating a post-quantum signature is a technical milestone. Making it verifiable, interoperable, and trustworthy over time is the real deployment challenge.

Thank you.