Build, boot, verify, publish

A publication contract for reusable machine images

Jeleel Muibi · HybridOps.Tech · Version 1.1 · August 2026

Abstract

A successful machine-image build proves that a construction procedure completed. Publication readiness requires additional evidence: the retained artifact must produce a runnable consumer, satisfy the configured readiness gate, resolve under a stable logical identity, and carry an explicit retirement path.

This paper introduces a publication contract between image construction and downstream infrastructure consumption. A declared image becomes a reusable dependency when the target decision is explicit, the retained image has passed its configured runtime exercise, the logical identity is published for consumers, and a defined retirement path exists. Construction and native provider state remain owned by their existing systems; the innovation is the control boundary that promotes a built artifact into an accepted infrastructure dependency.

The model was evaluated on Proxmox using Ubuntu 22.04, Ubuntu 24.04, Rocky Linux 9 and Windows 11 Enterprise. Five fresh observations per family completed construction, disposable clone-and-boot exercise, logical publication, reuse, retirement and idempotent retirement. All 20 observations completed the configured lifecycle. A separate extension completed three downstream VM handoffs that resolved published logical state rather than embedding a provider template identifier.

External review of v1.0 added two publication requirements that this edition makes explicit: verification must be bound to the artifact and inputs that produced it, and verification has a lifetime. A reusable image can remain present and bootable after the evidence that justified its publication has aged beyond the next consumer's policy.

The operational question is:

What must be proved before a completed image build is allowed to become a dependency for other infrastructure?

1. Build success and publication readiness are different states

Image construction answers an important question: did the declared build procedure complete? A consumer needs a second answer: is there now a stable, identified and tested image dependency that this environment is prepared to use?

Between those two states, an operator establishes:

1. whether the target identity is new, reusable or intentionally replaceable;
2. whether the image can produce a runnable consumer;
3. whether the required readiness observation appears after boot;
4. which logical identity consumers should resolve;
5. whether disposable validation resources were removed; and
6. how the retained dependency is retired later independently of ordinary workload deletion.

Making those decisions executable turns the image-factory handoff into a reviewable publication contract.

Established approaches and the remaining gap

Reproducible-build research addresses integrity and repeatability of software artifacts [1]. VM-image research has explored automated configuration, testing, inspection and semantics-aware image management [2-5]. Image builders and cloud or virtualisation platforms already provide construction, storage, cloning and native image metadata.

Those capabilities establish important parts of the lifecycle. The separate handoff problem is when an infrastructure runtime is allowed to promote a constructed image into a logical dependency that downstream operations may consume. That decision can require runtime exercise, target collision policy, publication semantics, consumer resolution and later retirement across systems without one native lifecycle.

The innovation introduced here is an executable publication contract for that handoff. The builder and image platform retain native construction and provider state authority; the publication contract governs the transition from completed artifact to accepted infrastructure dependency.

2. The publication lifecycle

The model separates six decisions.

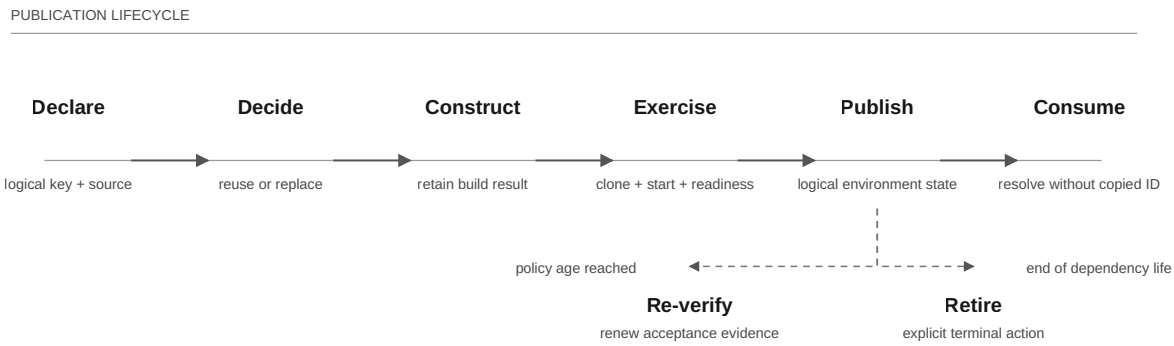

Figure 1. A declared template passes through target decision, construction, disposable runtime exercise, publication, consumption and explicit retirement.

Stage	Decision
Decide	Is the target absent, reusable or deliberately replaceable?
Construct	Did the selected image implementation produce the retained image?
Exercise	Can a disposable consumer clone/start and satisfy the configured readiness gate?
Publish	May the tested image identity become available under a logical environment key?
Consume	Can downstream infrastructure resolve that key without copying a provider identifier?
Retire	Can the retained dependency be removed explicitly and idempotently?

The acceptance gate is risk-adjusted to the consumer. A base operating-system image may require clone, boot and agent readiness; a more demanding image can require SSH, application or security checks before publication.

3. Make target identity a decision

Target identity is resolved before construction so reuse and destructive replacement remain explicit decisions.

The publication contract resolves one of three outcomes:

- the target is absent and construction can proceed;
- the target exists and replacement is not requested, so the retained image is reused and its logical state can be republished; or
- the target exists and replacement is explicitly requested, so the replacement action is visible before construction begins.

This separates idempotent reuse from destructive replacement.

4. Exercise the retained artifact through a consumer path

The acceptance gate tests the artifact in the form downstream infrastructure will consume.

In the measured implementation, required smoke creates a disposable full clone, starts it, waits for guest-agent IPv4 readiness, records available guest operating-system information and removes the validation VM.

The configured evidence boundary is:

Condition	Publication effect
Construction fails	Block
Required clone/start fails	Block
Required guest-agent readiness times out	Block
Guest OS identity differs from expectation	Record in measured release
Validation cleanup fails	Record; measured release treats cleanup as lifecycle evidence
Required gates complete	Publish logical image state

The word *verified* refers to these named runtime observations. Security posture, binary reproducibility and application correctness are separate evidence classes that can be added when the consumer contract requires them.

A practitioner review of v1.0 sharpened this boundary with an Azure Virtual Desktop example: an image can boot cleanly while FSLogix attachment, Kerberos, media optimisation or application-attach paths still fail. Those are real consumer failures at the workload/session boundary. The base-image publication gate therefore remains focused on the evidence class owned by the image contract, while higher-level workload verification remains with the workload lifecycle [10].

5. Publish logical state as the consumer contract

After acceptance, the retained image is published under a logical environment key. Downstream infrastructure resolves that state instead of embedding a provider template identifier directly in its own declaration.

Provider identifiers are implementation bindings. A logical image key is the consumer contract. If an accepted template is deliberately replaced, the consumer can continue to request the same logical image while the environment binding changes behind that contract.

This is an operation-scoped logical publication contract. Where an existing image platform already owns promotion channels, consumer selection, provenance and retirement, that system can remain the publication authority. HybridOps applies the publication contract where those responsibilities span builder output, provider procedure and downstream consumer configuration.

6. Publication evidence needs identity, provenance and freshness

A green clone-and-boot result answers a point-in-time runtime question. A shared dependency additionally needs three evidence properties.

Artifact identity. The consumer needs a way to know that the image being used later is the retained artifact, or an explicitly accepted successor, that passed the publication gate.

Provenance. A reusable image should be traceable to the relevant source, base image, build inputs, implementation version and acceptance run. That binding connects the accepted artifact to the inputs and procedure that produced it.

Verification freshness. An image can remain present and bootable while the basis for trusting it changes. A base CVE may be disclosed, a signing key may rotate, an embedded certificate or activation condition may expire, or policy may require re-verification after a defined interval. The relevant age is the last qualifying verification [10].

External review of v1.0 adds these requirements to the successor publication model. The controlled evaluation measured the runtime publication gate; the next evidence layer binds source, artifact identity and verification freshness into the same acceptance record.

The review also reinforces a design rule: keep time-sensitive, workload-specific state close to provisioning or session startup where its own freshness can be evaluated. Certificates, enrolment tokens, domain join and application layers are examples [10].

7. Retire or re-verify the dependency explicitly

A retained image has a lifecycle beyond any individual workload consumer.

Retirement resolves the declared image identity and removes the retained dependency through an explicit operation. An already absent image is treated as an idempotent terminal result.

The v1.0 review adds another transition: re-verify. A dependency that remains useful but whose acceptance evidence has exceeded policy age can pass the required gate again before further consumption.

The publication contract therefore has creation, re-verification and retirement paths. Acceptance creates a reusable dependency; policy can later require fresh evidence or explicit retirement.

8. Controlled evaluation

The primary controlled matrix used one frozen HybridOps release, one isolated Proxmox test environment and four operating-system families:

- Ubuntu 22.04;
- Ubuntu 24.04;
- Rocky Linux 9; and
- Windows 11 Enterprise.

Each family was observed five times. Every observation included a fresh build, required disposable clone-and-boot exercise, publication, existing-template reuse, explicit retirement and a second idempotent retirement.

A predefined fault block covers source integrity, identity collision, guest readiness, clone capacity, cleanup and guest-family mismatch. The primary matrix establishes configured lifecycle completion; the fault block remains the next controlled fault-evaluation layer.

A second extension created one fresh producer-to-consumer observation for Ubuntu 22.04, Ubuntu 24.04 and Rocky Linux 9. Each consumer resolved published producer state without a hardcoded template VMID, passed required SSH readiness and retired cleanly.

9. Results

All 20 primary observations completed the configured lifecycle.

Template	Completed	Median construction	Median interval after construction	Median total to published state	Residual validation VMs
Ubuntu 22.04	5/5	8.73 min	0.84 min	9.56 min	0
Ubuntu 24.04	5/5	8.34 min	0.72 min	9.11 min	0
Rocky Linux 9	5/5	6.33 min	0.69 min	7.04 min	0
Windows 11 Enterprise	5/5	11.37 min	1.78 min	13.25 min	0

Across the observations, the highest sampled thin-pool use was 94.45% against a 96.50% stop boundary. Every observation returned to the recorded 90.65% baseline. The final provider scan found no VM in the reserved template or validation ranges.

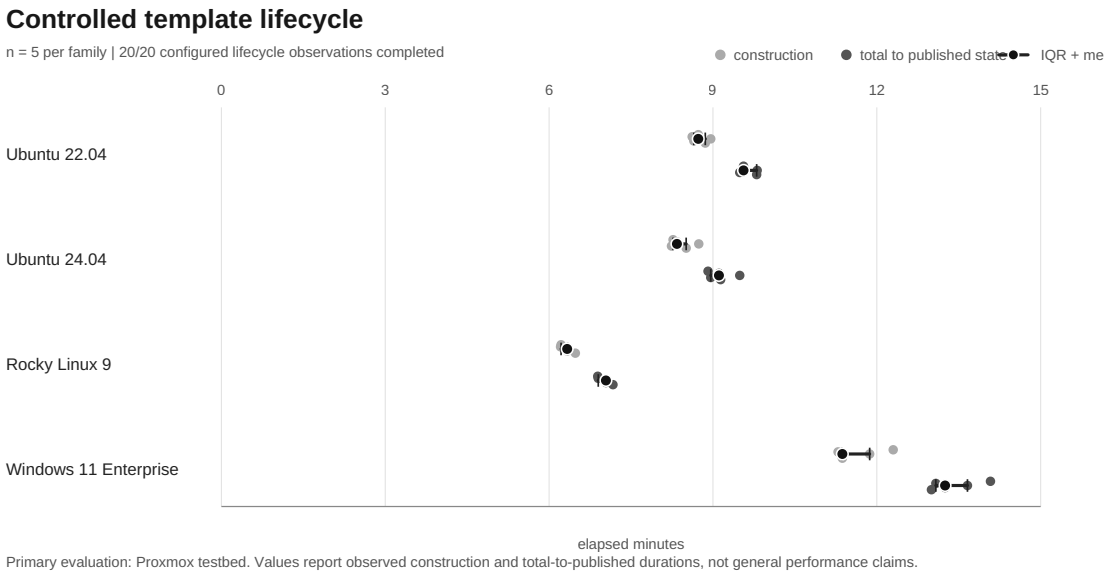

Figure 2. Individual construction and total-to-published-state durations for the 20 controlled observations, with inclusive interquartile ranges and medians.

The measured interval beyond construction is the explicit cost of the additional publication boundary in this testbed. Its family median ranged from 0.69 to 1.78 minutes, providing a concrete basis for evaluating the gate against its operational benefit.

All three downstream extension consumers resolved producer state and reached required SSH readiness before clean consumer and producer retirement.

Template	State-resolved consumer	Required SSH readiness	Clean retirement	Residual resources
Ubuntu 22.04	1/1	1/1	1/1	0
Ubuntu 24.04	1/1	1/1	1/1	0

Template	State-resolved consumer	Required SSH readiness	Clean retirement	Residual resources
Rocky Linux 9	1/1	1/1	1/1	0

The evidence retains observed anomalies: a primary-IP synchronisation conflict in the Ubuntu 22.04 extension, an early storage-sampler exit in the Rocky Linux observation, and inconsistent Windows 10/11 identity fields in the Windows guest-agent record. These remain part of the evaluation record.

10. What the evaluation establishes

The controlled observations establish that, in the measured Proxmox environment, the implemented contract can move a declared image through target decision, construction, required runtime exercise, logical publication, downstream state resolution and explicit retirement while retaining structured evidence of those transitions.

The next evidence classes are:

- security posture and vulnerability status;
- strict operating-system identity acceptance;
- controlled fault-matrix execution;
- binary reproducibility between independent builds;
- signed artifact/provenance binding;
- verification-age enforcement;
- equivalent evaluation on additional hypervisors or cloud image services; and
- independent external replication.

These tests extend the evidence model beyond the completed runtime-publication result.

11. HybridOps implementation boundary

HybridOps is the open-source implementation used to exercise this model. The image builder remains authoritative for construction and the virtualisation platform remains authoritative for stored templates, clones and provider tasks.

HybridOps introduces the publication contract across those systems: target policy, required runtime exercise, logical state publication, downstream resolution and explicit retirement become one lifecycle rather than separate operator conventions.

The successor model adds verification identity, provenance and freshness as explicit requirements for future acceptance evidence while keeping the controlled runtime results as their own measured evidence class.

12. Practitioner review question

For an image your infrastructure depends on, what evidence should expire or force re-verification before the image can be consumed again?

A reviewer can also identify cases where an existing image platform already owns the complete acceptance, promotion, freshness and retirement lifecycle, which helps define the natural publication boundary.

References

1. Lamb, C. and Zacchiroli, S. (2022). Reproducible Builds: Increasing the Integrity of Software Supply Chains. *IEEE Software, 39*(2), 62-70.

2. Spichkova, M. et al. (2020). VM2: Automated security configuration and testing of virtual machine images. *Procedia Computer Science, 176*, 3610-3617.
3. Yamato, Y. (2015). Automatic verification technology of software patches for user virtual environments on IaaS cloud. *Journal of Cloud Computing, 4*, article 4.
4. Suneja, S. et al. (2017). Safe Inspection of Live Virtual Machines. *VEE 2017*, 97-111.
5. Saurabh, N., Remmers, J., Kimovski, D., Prodan, R. and Barbosa, J. G. (2019). Semantics-aware virtual machine image management in IaaS clouds. *2019 IEEE 33rd International Parallel and Distributed Processing Symposium (IPDPS)*, 418-427.
6. HybridOps Core, open-source implementation.
7. HybridOps Core, template-image module.
8. HybridOps, C03 controlled lifecycle evidence.
9. HybridOps, C04 producer-consumer evidence.
10. HybridOps Core issue #270, external technical review of Build, Boot, Verify, Publish v1.0.