Authority before automation

An executable authority model for infrastructure intent

Jeleel Muibi · HybridOps.Tech
Technical paper · Version 1.0
First published May 2026 · revised 8 August 2026

Abstract

Infrastructure automation can execute a change quickly while still acting on the wrong version of intent. An address may exist in a spreadsheet, a virtual machine definition in code, a different value in a state file and another value on the running platform. Adding another automation task does not resolve which record should prevail.

HybridOps addresses this as an authority problem. Each governed fact has a declared owner, and intended state, implementation specification, execution binding and observed condition remain separate records. Before an operation uses a governed input, the runtime checks that the operation and its dependent step agree on the required authority and that the selected authority is ready. Observation cannot silently promote itself into intent.

The public implementation exercises this model through a bounded bootstrap and authority-transfer path. Initial infrastructure is created from limited bootstrap input. Once the intended-state service is established and seeded, downstream provisioning requires that authority. A mismatch, missing state or failed required live check stops the dependent operation before infrastructure is changed.

The proposed rule is narrow. Before an automated operation consumes a governed input, it must identify the record authorised to supply that input and establish that the authority is ready.

1. The problem is not a lack of data

Most infrastructure estates already contain the information needed to operate them. The difficulty is that the information is repeated across configuration repositories, cloud consoles, virtualisation platforms, inventory systems, runbooks and local files.

The copies do not have equal meaning. A cloud API can report that an instance exists, but it cannot decide whether that instance still belongs in the intended design. A state file can bind an automation run to a provider object, but it should not become the organisation's IP address plan. A monitoring check can report that a service is unavailable, but it does not define the approved service topology.

When ownership is unclear, an operator must infer intent from whichever record looks newest. That creates three common risks:

- an observed value silently replaces an approved design decision;
- stale derived data is accepted because an earlier run succeeded; and
- two automation paths update the same fact under different assumptions.

The first requirement is therefore an authority map, not another synchronisation job.

2. Intended, execution and observed state

The distinction is established in network-management standards. RFC 8342 separates intended configuration from operational state and recognises that the two may differ because of propagation time, missing resources, system behaviour or failed application. RFC 9144 describes these as different viewpoints in the lifecycle of management data.

NetBox follows the same principle for infrastructure modelling. It represents intended state rather than treating discovery as authority. The implementation evaluated in this paper uses it for governed addressing, topology and inventory, but the authority model is not defined by that product.

Execution state has a narrower responsibility. It can bind a declared object to a provider resource and expose drift, but it cannot decide whether an external change should be accepted into organisational intent or reverted. Detection does not make that policy decision.

HybridOps uses four distinct records:

Record	Governing question	Responsible record
Declared intent	What should exist, where and under which relationships?	The designated authority for the governed fact
Implementation specification	How should the declared outcome be produced?	The versioned operation contract
Execution binding	Which provider objects belong to this run and environment?	Runtime and native infrastructure state
Observed condition	What exists now, and is it usable?	Live provider and workload checks

These records can be compared, but they should not overwrite one another without an explicit transition.

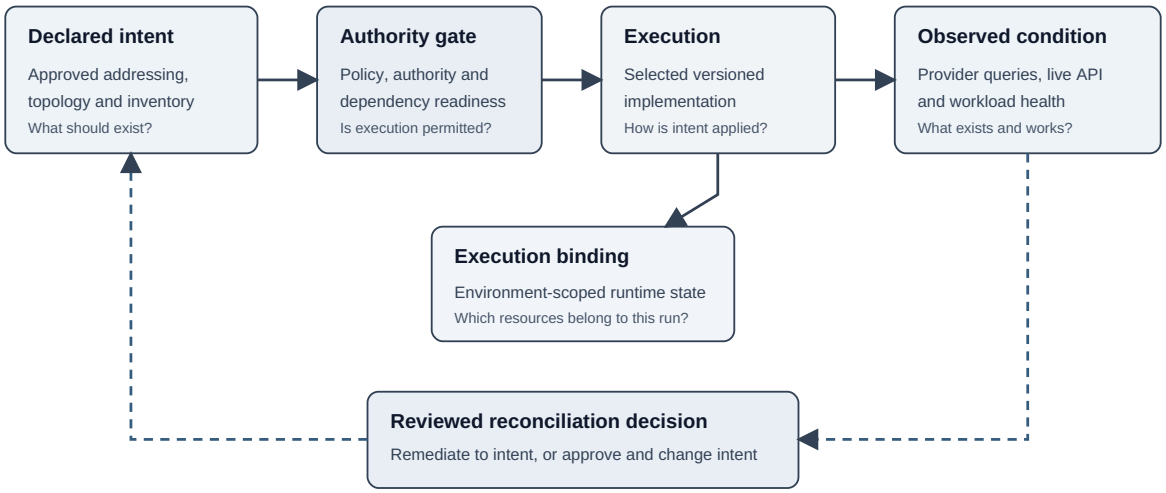

Figure 1. HybridOps separates declared intent, execution binding and observed condition through an explicit authority contract.

3. The HybridOps authority gate

A HybridOps operation declares policy for the whole lifecycle and requirements for each dependent step. A step that consumes governed addressing must declare both the addressing mode and the authority it requires. The enclosing operation must make the same authority decision. A mismatch is an invalid operation, not a default to be guessed at runtime.

In the implemented path, Core checks that:

1. the blueprint policy and step contract agree;
2. the configured authority environment is valid;
3. the intended-state service has successful runtime state;
4. any required upstream module state is available; and
5. when required, its live API responds using the resolved credentials.

A failed condition blocks the step. It is not reduced to a warning and allowed to continue with guessed addressing.

The authority can live in a shared runtime environment and serve several workload environments without mixing their execution state. A deliberate override can select another authority environment, but the selection remains explicit.

The same rule applies after provisioning. When publication to the declared authority is required, the export must succeed, contain usable records and complete its synchronisation. Resource creation is not reported as a complete governed operation when that required publication fails.

4. Resolving the bootstrap problem

There is an unavoidable first-run question: how can an intended-state service govern infrastructure before that service exists?

HybridOps makes the circular dependency explicit. The implemented onprem/bootstrap-netbox@v1 path uses limited static input to establish the minimum foundation:

1. prepare the software-defined network and base image;
2. create the database and NetBox hosts on declared bootstrap addresses;
3. configure PostgreSQL and NetBox; and
4. seed the foundation datasets.

This is a transition, not a permanent second authority. Once the service is ready, the onprem/authoritative-foundation@v1 operation changes the authority policy. Downstream virtual machines and services consume governed IPAM, and the intended-state service becomes a required gate.

The separation matters. Calling a bootstrap spreadsheet or static file a source of truth would preserve two competing owners. Treating it as a bounded bootstrap input provides a clear point at which authority transfers.

5. Reconciliation without silent ownership changes

Continuous reconciliation is useful only when the desired state is clear. OpenGitOps describes a desired system as declarative, versioned, automatically pulled and continuously reconciled. Kubernetes controllers similarly operate control loops that move current state toward desired state, while warning that multiple controllers need clear ownership boundaries.

HybridOps applies the same discipline across a mixed estate:

- an authoritative foundation step is allowed to rerun even when earlier module state is successful, so stale success cannot conceal authority drift;
- derived inputs can be refreshed from authoritative upstream outputs;
- stored input fingerprints expose a changed declaration;
- live checks can distinguish a successful historical run from a currently reachable authority; and
- provider and workload checks establish what was applied and whether it is usable.

Reconciliation is not a licence to copy the live estate back into the intended-state authority. Unexpected operational state requires a decision:

1. correct the implementation so it returns to declared intent; or
2. review and approve the external change, then update the authority and reconcile from the new declaration.

That preserves a record of why intent changed. In the implemented case, custom validation, protection rules and change history further constrain changes to the authoritative dataset.

6. Evidence from the implemented authority transition

The public authoritative-foundation case exercises the model across a virtualised on-premises foundation. The relevant evidence is not the presence of an inventory product. It is the behaviour at each authority boundary.

Boundary	Implemented evidence	Required behaviour
Bootstrap	The bootstrap operation declares no IPAM authority and uses bounded static inputs	Bootstrap data cannot silently remain authoritative
Transfer	The authoritative operation declares governed IPAM and dependent steps require the same authority	A policy or step mismatch is rejected
Readiness	Stored service state and an optional live API probe establish availability	Missing or unusable authority blocks dependent work
Publication	Selected infrastructure outcomes are exported under a strict publication hook	Failed required publication prevents governed completion
Observation	Provider and workload checks report the running result separately	Observation does not rewrite intended state

The implementation establishes the foundation service, seeds its intended dataset and then provisions downstream virtual machines from governed addressing. The resulting prefixes, inventory and service relationships are visible in the published walkthrough. Source contracts and runtime checks are linked in the references.

The individual mechanisms are established capabilities. HybridOps owns the decision about whether the wider operation may consume or publish governed state. It enforces declared authority, an explicit bootstrap transition, readiness before consumption, controlled publication after execution and separate observation of the running result.

7. Governance and operational review

NIST SP 800-128 places baselines, change control and configuration monitoring within security-focused configuration management. NIST SP 800-53 Revision 5 includes corresponding controls for configuration baselines, settings and system component inventory.

Organisational approval remains outside the runtime. HybridOps provides enforcement points where an approved authority policy becomes executable:

- schema validation before a blueprint is accepted;
- authority and dependency gates before a change begins;
- environment-scoped state and run records during execution;
- required publication of selected infrastructure outcomes;
- workload verification after configuration; and
- guarded teardown for lifecycle completion.

The practical test is whether an operator can answer four questions after a change:

1. Which record authorised the addressing and inventory decision?
2. Which versioned operation performed the change?
3. Which provider resources were bound to the run?
4. Which checks established the resulting operational condition?

If the answers come from four unrelated guesses, the estate is automated but not governed.

8. Request for technical review

Three questions define the requested review:

1. Does authority per governed fact, rather than one global source of truth, reflect the control boundary needed in a hybrid estate?
2. Do the bootstrap transfer, readiness gate and required publication provide sufficient protection against conflicting authority?
3. Which brownfield or emergency-change case would most usefully test this model next?

A short technical assessment or counterexample is sufficient.

References

1. IETF, RFC 8342: Network Management Datastore Architecture, 2018.
2. IETF, RFC 9144: Comparison of NMDA Datastores, 2021.
3. NetBox Labs, Introduction to NetBox.
4. NetBox Labs, Custom validation.
5. NetBox Labs, Data protection rules.
6. NIST, SP 800-128: Security-Focused Configuration Management, 2011, updated 2019.
7. NIST, SP 800-53 Revision 5: Security and Privacy Controls, 2020.
8. OpenGitOps, GitOps principles.
9. Kubernetes, Controllers.
10. HybridOps.Tech, HybridOps.Core.
11. HybridOps.Tech, authoritative foundation operation.
12. HybridOps.Tech, bootstrap authority operation.
13. HybridOps.Tech, source-of-truth architecture decision.
14. HybridOps.Tech, authoritative foundation case.