Orchestrate and roll back connector hosts

Use this guide to reconcile one connector instance through an external workload platform without giving that platform connector-registry administrator access. The procedure derives a deterministic plan from a verified admission package, an

SECTION	Deploy and Operate
TYPE	Guide
PROTOCOL	AIP 1.0
SOURCE	docs/guides/connector-orchestration.md

The contract is implemented by `aip-connector-orchestration` at source revision `97be86e9efedf07ecf1783b03800f683f107fb04`. It is product-neutral and applies to standalone Cal.diy, Hermes Agent, Chatwoot, Dify, CrewAI, and Twenty hosts.

AIP does not contain a Kubernetes, Nomad, ECS, Docker, or other platform executor. `aipctl connector orchestration plan`, `sign`, and `verify` produce and validate JSON; none of them starts a process or writes connector-registry state. A deployment-owned adapter must implement the verified operations and the safety checks described here.

Keep orchestration outside registry authority

The orchestration contract separates four authorities:

Authority	Owns	Must not own
Release and admission	Signed package, immutable version, evidence, instances, and pre-provisioned replicas	Workload-platform credentials
Orchestration signer	Reviewed deployment intent and signed deterministic plan	Registry administrator credentials or provider secrets
External executor	Platform workload creation, observation, graceful termination, and operation journal	Catalog mutation or unsigned replica identity changes
Host lifecycle service	Signed register, heartbeat, drain, and offline transitions	Admission packages or platform scheduling

The plan contains an opaque `secret_provider_ref`, never secret material. The executor maps that reference to an approved platform mount without adding the resolved bytes to plans, labels, command lines, observations, or journals.

Routing bindings are also outside this contract. Orchestration can make an already admitted replica process exist, but only a separately admitted enabled binding makes it eligible for a tenant capability.

Know the reconciliation inputs

One reconciliation cycle uses these exact inputs:

Input	Purpose
Signed admission package	Proves package identity, active immutable version, instance, replica identities, digest, and evidence
Admission trust policy	Verifies the package and all required evidence roles

Input	Purpose
Deployment intent	Declares one instance, monotonic generation, desired replica IDs, rollout budget, and time window
Observed replicas	Reports the external platform's exact state for that instance
Orchestration trust policy	Defines authorized plan signers and hard reconciliation limits

The reconciler derives `ReplicaTarget` values from the verified package. An executor cannot inject a hidden target into a plan: the target map must exactly equal the intent's desired replica set, and verification recomputes the operation list from the embedded intent, targets, and observations.

Each target fixes:

- replica, instance, and version IDs;
- lowercase SHA-256 artifact digest rather than a mutable tag;
- HTTPS native endpoint ending exactly in `/aip/v1/messages`;
- peer principal and `did:key`;
- trust domain and topology;
- instance configuration revision and opaque secret-provider reference;
- admitted replica capacity.

The orchestration validator does not allow the host's loopback HTTP development exception. A target must use HTTPS and contain no URL credentials, query, or fragment.

Define the executor trust policy

Install an executor-local policy before accepting plans:

```
JSON
{
  "trusted_signer_dids": ["did:key:<trusted-orchestration-key>"],
  "limits": {
    "max_replicas_per_instance": 1000,
    "max_observations": 2000,
    "max_operations": 2000,
    "max_plan_ttl_seconds": 300,
    "max_clock_skew_seconds": 30
  }
}
```

The values shown are implementation defaults, not deployment sizing advice. All collection limits must be non-zero. Plan TTL can be at most 86,400 seconds, and accepted clock lead can be at most 300 seconds. Set lower bounds where the platform and review process can support them.

The signer key must derive one of `trusted_signer_dids`. Keep the 32-byte Ed25519 seed in a 64-hex-character owner-only file. The executor should receive only the trust root, not the signing seed.

Build a complete platform observation

Read the platform and connector registry immediately before planning. Include every process observed for the selected instance, not only the desired set. Each observation contains:

Field	Meaning
`replica_id`	Concrete pre-provisioned or extra observed replica
`instance_id`	The one instance being reconciled
`version_id`	Version reported by the immutable process artifact

Field	Meaning
`artifact_digest`	Digest reported by the platform runtime
`generation`	Deployment generation applied by the platform
`phase`	`starting`, `ready`, `draining`, `stopped`, or `failed`

Report `ready` only when the process serves and its connector-registry lease is ready. A Kubernetes-style pod readiness bit alone is not sufficient. Report `draining` while the common lifecycle completes pinned work, `stopped` when the process is absent, and `failed` for a terminal or restartable platform failure.

Observations must have unique replica IDs, a non-zero generation, and the exact intent instance. An observed generation newer than the proposed intent fences the intent as stale. An absent desired replica may be omitted; the reconciler can then produce `start` within budget.

Do not conceal an extra replica. The reconciler needs it to produce a bounded drain or stop operation.

Create the deployment intent

Use schema `aip.connector-orchestration/v1`. The intent contains:

Field	Rule
`generation`	Non-zero and monotonically newer than the platform generation being changed
`rollback_of_generation`	Omit for normal rollout; for rollback name the earlier generation being restored
`package_id`	Stable signed admission package stream ID; 1–256 bytes
`package_revision`	Exact non-zero verified package revision
`package_digest`	Exact lowercase canonical SHA-256 digest returned by admission verification
`instance_id`	One instance contained in that package
`desired_replicas`	Exact set of pre-provisioned replica IDs that should run
`max_surge`	Temporary active processes allowed above the desired count
`max_unavailable`	Desired replicas allowed to be unavailable during replacement
`issued_at`, `expires_at`	Reviewed time window within the policy TTL

`max_unavailable` cannot exceed the desired count. `max_surge` cannot exceed the per-instance limit, and desired plus surge must fit that limit. An empty desired set is the explicit scale-to-zero intent.

The verified connector type and instance must be enabled and the version must be active before a non-empty desired set can start. Disabled types and instances may only scale to zero.

Retain the exact intent as a change artifact. Do not silently refresh its generation, package digest, desired set, budget, or time window after review.

Derive and review the next operation batch

Plan without signing or writing:

```
SH
aipctl connector orchestration plan \
  --package admission-signed.json \
  --admission-trust-policy admission-trust-policy.json \
  --intent deployment-intent.json \
  --observed platform-observed.json \
  --orchestration-policy orchestration-policy.json > orchestration-plan.json
```

Review package coordinates, instance, generation, desired set, target digests, observed snapshot, rollout bounds, and every operation. The reconciler emits only four operation types:

Operation	Reconciler condition	Executor effect
`start`	Desired replica is absent or stopped and surge budget exists	Create the exact immutable target.
`restart`	Desired replica is failed	Restart the same target without changing identity or digest.
`drain`	Ready old or old-generation replica can leave within unavailability budget	Request common graceful host shutdown and verify signed drain state.
`stop`	Replica is stopped or failed; an old-generation desired replica is starting or draining; or a non-desired replica is draining or starting during scale-to-zero	Remove only after the operation's full safety precondition is true.

Operations are canonically sorted as start/restart, then drain, then stop, with replica ID as the stable tie-breaker. This order is not permission to run every operation concurrently. Execute in the signed order and obey the rollout budget and per-operation preconditions.

For a replacement with surge capacity, the first cycle normally starts the new replica and does not drain the old one. A later cycle can drain the old replica only after fresh observation reports enough ready capacity.

An empty operation list means the observed deployment is reconciled with this intent.

Sign the deterministic plan

Derive and sign the exact batch with a trusted orchestration authority:

```
SH
aipctl connector orchestration sign \
  --package admission-signed.json \
  --admission-trust-policy admission-trust-policy.json \
  --intent deployment-intent.json \
  --observed platform-observed.json \
  --orchestration-policy orchestration-policy.json \
  --signing-seed-file /run/operator/orchestration-signing-seed.hex \
  --signer-identity deployment-controller > orchestration-signed.json
```

`sign` re-derives the plan, requires its signing DID to be trusted by the executor policy, signs canonical plan JSON, and verifies the result before printing it. The signer identity must contain 1–512 bytes.

Treat the signed file as immutable. A modified generation, target, observation, operation, order, or time window invalidates its signature or deterministic semantic check.

Fence execution against fresh state

Read the platform again immediately before the first operation and write a new bounded observation file. Verify the signature, limits, time window, deterministic operations, and exact snapshot:

```
SH
aipctl connector orchestration verify \
  --plan orchestration-signed.json \
  --current-observed platform-current.json \
  --orchestration-policy orchestration-policy.json
```

Require `status: verified`, `write_performed: false`, the expected package and generation, `snapshot_fenced: true`, and one `orchop_...` ID per operation. A changed phase, generation, version, digest, missing replica, or newly observed replica invalidates the complete batch. Reconcile and sign a new plan; do not edit the stale plan.

Journal before executing

Before the first platform effect, durably store:

- canonical signed-plan bytes and digest;

- signer DID and identity;
- package ID, revision, and digest;
- instance and generation;
- fresh observed snapshot and verification time;
- ordered operation IDs and payloads;
- per-operation state, attempts, platform resource identity, and result;
- stop, rollback, and escalation decisions.

An operation ID is a SHA-256-derived `orchop_` value bound to the complete plan digest, operation index, replica ID, and payload. Use it as the platform adapter's idempotency key. Once the first operation begins, resume only this exact signed plan; do not reject it merely because execution itself changed the original observed snapshot.

The executor must be safe after response loss. Check the journal and platform state before retrying a start, restart, drain, or stop. Never allocate a new replica ID or select a mutable tag during retry.

Enforce operation-specific safety

For `start` and `restart`:

1. resolve only the target's opaque secret reference;
2. require the exact artifact digest from the signed target;
3. apply the exact replica ID, instance, version, endpoint, principal, DID,

trust domain, topology, configuration revision, and capacity;

4. wait for common host registration and `/ready`;
5. compare the registered identity with the signed target;
6. record the platform and lease evidence.

For `drain`, request the platform's graceful termination path. The host itself must sign its lifecycle transition; the executor must not impersonate the host or write registry state. Preserve the process while pinned work completes.

For `stop`, treat the operation contract's phrase “fully drained” as a hard precondition. The current reconciler can list `stop` when an observation says `draining`; that phase alone does not prove zero in-flight provider work.

Begin and journal the stop operation, but defer destructive platform removal until host lifecycle evidence shows the process has completed drain and is offline or absent. A failed process may be removed only after its lease and durable-work state are contained.

If a precondition cannot be established, leave the journal entry pending and escalate. Do not reinterpret a signed `stop` as permission for a blind kill.

Reconcile until no operations remain

After the signed batch completes:

1. capture a new complete observation for the same instance;
2. keep the reviewed generation and desired set unchanged;
3. issue a newly reviewed time window if the earlier intent expired;
4. derive and sign the next deterministic batch;
5. verify against another fresh snapshot;

6. journal and execute that exact batch;
7. repeat until the signed plan contains no operations.

A generation refresh of the same immutable replica normally progresses through drain, stop, and start across observations. A rollout to new replica IDs starts ready replacements before old replicas can drain when the surge and unavailability budget require it. Scale-to-zero uses an empty desired set and drains ready replicas before removing them.

Completion also requires registry readiness and tenant-scoped discovery where the deployment is intended to receive traffic. An empty operation list alone does not enable a binding or prove provider qualification.

Roll back with a new generation

Rollback is a new desired-state transition, not reversal of journal entries:

1. select a still-trusted admission package whose connector type and instance

are enabled and whose immutable version is active;

2. use only replica targets pre-provisioned in that verified package;
3. create a generation newer than every observed platform generation;
4. set `rollback_of_generation` to the earlier known-good generation being

restored;

5. set the desired replica set and bounded surge/unavailability values;
6. plan, sign, fresh-snapshot verify, journal, and execute normally;
7. reconcile until the rollback plan is empty;
8. verify bindings, routes, host readiness, and provider behavior separately.

Do not decrement generation. Do not reuse a current replica ID for an older version or digest. If an old version is revoked, this orchestration contract cannot reactivate or start it; admit an acceptable immutable version through the registry process instead.

Rollback does not change tenant bindings, rewrite existing route assignments, undo provider mutations, or rotate credentials. Coordinate those owning procedures explicitly and retain ambiguous-effect evidence.

Stop conditions

Discard the proposed batch and reconcile again when:

- the signed admission package or its evidence no longer verifies;
- package coordinates or digest differ from the intent;
- a desired replica is absent from the package;
- a desired observed replica reports a different version or digest;
- platform state changed before the first operation;
- an observation belongs to another instance, is duplicated, or has a newer

generation;

- the plan expired or exceeds executor limits;
- the signer is untrusted or the signed operation list is not deterministic;
- a start would use a mutable artifact or unreviewed secret mapping;

- a drain would violate minimum ready capacity;
- a stop lacks fully drained, offline, or contained-failure evidence;
- the journal cannot prove whether an operation began.

For admission and initial target provisioning, use Deploy the connector fleet. Follow Operate the connector host lifecycle for signed drain and recovery, Rotate connector credentials for secret and binding changes, Deploy AIP in production for authority separation, and Identity and trust for signing and opaque-reference boundaries.