Deploy the Cal.diy connector

Use this guide to deploy one aip-host-cal-diy instance for one Cal.diy account, prove that its admitted identity and provider connection are ready, and introduce it to tenant traffic without creating a second account owner.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/cal-diy/operations/deployment.md

Complete the common connector fleet deployment first. This page adds the Cal.diy artifact, account, authentication, network, and rollout decisions. It does not turn the repository qualification Compose file into a production topology.

Preserve the deployment boundary

The standalone process owns one stable Cal.diy account boundary. Every replica of that instance uses the same admitted connector instance, external account, credential policy, and durable runtime state. A replacement process receives a new replica identity; it does not create a second instance for the same account.

Applications send signed actions to product-neutral `aipd`. The gateway routes an admitted action to the host's `/aip/v1/messages` endpoint. Applications do not call the host or the Cal.diy API directly.

The source-owned container image builds package and binary `aip-host-cal-diy`, runs as user `10001`, exposes process port `8081`, and has a container liveness check on `/health`. Pin the released image by immutable digest. A mutable image tag is not an admission identity.

1. Freeze one deployment record

Record these values before preparing the host or an admission package:

Value	Cal.diy rule
Connector type and version	Match the admitted Cal.diy manifest and immutable artifact
Instance ID	Represents one tenant-owned Cal.diy account
Replica ID	Identifies one concrete host process and is never reused concurrently
Tenant and membership	Match the tenant binding admitted for this instance
External account	Set the stable account ID and system `cal_diy`
Public endpoint	End in `/aip/v1/messages` and resolve to this replica
Host identity	Match the pre-provisioned replica DID and signing seed
Artifact digest	Match the bytes running in the container
Configuration revision	Increase when the deployment-owned configuration changes
Credential revision	Identify the credential revision accepted for new assignments

Use a distinct host signing seed and replica ID for every replica. Keep the connector instance and external account stable across a normal replacement. Changing account ownership requires a new reviewed instance boundary.

2. Prepare the runtime and network

Provision a durable PostgreSQL database for this connector boundary. Do not use the registry database or container filesystem for runtime state. A replacement must reconnect to the retained database before it can recover work and register.

Allow only the required flows:

Direction	Peer and route	Purpose
Inbound	`aipd` to `/aip/v1/messages`	Signed manifest-pinned AIP traffic
Inbound, operator network	`/health`, `/ready`, `/metrics`, `/aip/v1/manifest`	Liveness, readiness, metrics, and contract inspection
Inbound, optional	Cal.diy to `/webhooks/cal-diy/{subscription_id}`	Authenticated provider event ingress
Outbound	Connector lifecycle control endpoint	Signed register, heartbeat, drain, and offline transitions
Outbound	Host PostgreSQL	Durable actions, events, profile state, and recovery
Outbound	`AIPD_CAL_DIY_BASE_URL`	Cal.diy API calls and authenticated health probe
Outbound, when configured	Central callback endpoint	Stream callbacks through the pinned route
Outbound, required for webhooks	Central connector-event endpoint	Durable publication of accepted provider events

Terminate external TLS at a reviewed ingress or service proxy. Pin the gateway and lifecycle identities and the required CA roots in the host. Do not expose the observation routes to an untrusted network. Preserve raw webhook request bytes and the required signature and version headers through every proxy.

3. Stage configuration and secrets

Set the common host contract from Connector host configuration. Then set the Cal.diy values:

Setting	Requirement
`AIPD_CAL_DIY_BASE_URL`	Absolute provider origin owned by this deployment
`AIPD_CAL_DIY_ACCOUNT_ID`	Stable account boundary matching admission
`AIPD_CAL_DIY_BEARER_TOKEN_FILE`	Bearer credential file when bearer authentication is selected
`AIPD_CAL_DIY_OAUTH_CLIENT_ID`	Public client ID when OAuth client credentials are selected
`AIPD_CAL_DIY_OAUTH_CLIENT_SECRET_FILE`	Secret file paired with the OAuth client ID
`AIPD_CAL_DIY_MAX_RESPONSE_BYTES`	Optional positive provider-response limit
`AIPD_CAL_DIY_WEBHOOK_SECRET_FILES`	Optional comma-separated `SUBSCRIPTION_ID=SECRET_FILE` mappings
`AIPD_CAL_DIY_WEBHOOK_SUBSCRIBER_PREFIXES`	Optional approved HTTPS destination prefixes for webhook creation

Select exactly one authentication mode: one bearer-token file, or one complete OAuth client-ID and client-secret-file pair. The process rejects an incomplete or mixed selection before it serves traffic.

Mount database URLs, the host signing seed, provider credentials, and webhook secrets as owner-controlled files. Mount public DIDs, CA certificates, endpoint data, and the reloadable credential-revision policy separately. Keep secret values out of environment variables, image layers, logs, and admission packages.

If webhook secret mappings are present, configure the central event endpoint. The host refuses to start webhook ingress without its signed event publisher. Review the complete ingress contract in Webhook security and replay.

4. Harden and admit the artifact

Run the container with a read-only root filesystem, user `10001`, all Linux capabilities dropped, `no-new-privileges`, and a small `noexec,nosuid` temporary filesystem. Mount only the state, trust, and secret files required by this instance. These controls mirror the source-owned qualification composition; adapt their storage and ingress wiring to the production platform.

Verify the image digest, manifest digest, connector version, and implementation support map before signing admission. Pre-provision the instance and replica, then apply the first package with the intended tenant capability binding disabled. Do not let a host process create its own registry identity.

5. Start and prove readiness

Start the exact admitted artifact. The host performs this sequence before it serves AIP traffic:

1. validates common and Cal.diy configuration and reads bounded secret files;
2. creates the fixed Cal.diy connector and its 81-capability manifest;
3. opens PostgreSQL and recovers durable runtime state;
4. constructs optional webhook replay and event-publication state;
5. verifies storage and the authenticated Cal.diy profile endpoint;
6. registers the pre-provisioned replica and applies its lifecycle lease;
7. starts heartbeat renewal and the HTTP server.

Use `/health` only to determine whether the process can answer. The image's built-in health check does not prove provider, storage, or lease readiness. Require `/ready` to return success and confirm all four conditions:

- `draining` is false;
- `lease_valid` is true;
- `runtime.ready` is true;
- `connector.ready` is true.

The Cal.diy connector probe performs an authenticated profile read. A successful readiness response therefore proves the configured credential can reach that endpoint at that moment. It does not qualify the other 80 capabilities or prove that mutations will succeed.

Compare the served manifest and running artifact with the admission record. Confirm that the gateway sees the expected replica, endpoint, DID, topology, lease, and zero active assignments before enabling discovery.

6. Introduce one safe path

Apply a higher signed package revision that enables only the reviewed tenant binding. Through the authenticated gateway boundary:

1. confirm that the tenant can discover the intended capability;
2. confirm that another tenant cannot discover it;
3. call `cap:cal_diy:profile.get` with a fresh action ID;
4. retain the route assignment, result, receipts, provider request ID, and

redacted host evidence;

5. verify that the intended Cal.diy account produced the result.

Do not begin with booking, event-type, webhook, or other mutations. Introduce those groups only after their approval, idempotency, compensation, and account scope have been reviewed. Follow Authentication and tenant routing for the caller boundary.

7. Roll out a replacement

Keep one logical owner for the Cal.diy account throughout the rollout. If the platform overlaps replicas, they must use distinct replica identities under the same connector instance, the same external account, and shared durable state.

For a normal replacement:

1. admit the new immutable version and a new replica identity;
2. start one new replica with the intended binding still controlled;
3. require registration, manifest parity, a current lease, and `/ready`;
4. route only the reviewed canary scope supported by the fleet policy;
5. compare profile-read, error, latency, lease, and provider evidence;
6. pause new mutations and reconcile every uncertain or pinned action;
7. terminate the old replica through its normal shutdown signal;
8. require drain completion and the registry `offline` transition;
9. expand the new version only after retained evidence passes review.

`SIGTERM` and `Ctrl-C` invoke the source-owned graceful path. The host marks itself draining, rejects new actions with `connector_host.not_ready`, waits for in-flight work up to the configured drain timeout, stops heartbeat and serving, and requests the offline transition. An orchestrator kill timeout must exceed the host drain timeout plus network margin.

Roll back without losing work

Disable the affected binding in a higher signed package revision before restoring an earlier version. Confirm that new discovery and assignments have stopped, but do not assume this cancels provider work already in progress.

Reconcile pinned actions and unknown outcomes against the replica that owns their route. Drain the rejected version normally, require it to become offline, and retain its PostgreSQL state, outbox, replay records, logs, and audit evidence. Restore the prior immutable artifact and its matching admission and configuration as one reviewed unit. Never retry an uncertain mutation through a different replica merely because rollback completed.

Resolve deployment failures

Symptom	Decision
Process exits before registration	Check the exclusive authentication mode, bounded secret files, URLs, identity fields, and webhook event endpoint
Registration is rejected	Compare type, version, instance, replica, tenant, membership, endpoint, DID, artifact digest, manifest digest, and configuration revision with admission
`/health` succeeds and `/ready` fails	Inspect `lease_valid`, `draining`, runtime storage detail, and the authenticated Cal.diy profile probe
Provider probe fails	Verify the provider origin, CA trust, account boundary, selected credential mode, credential revision, and provider availability
Lease expires or heartbeat failures rise	Remove the replica from new traffic and inspect lifecycle reachability, signed control responses, clocks, and registry state
Webhook-enabled host will not start	Require the central event endpoint and validate every `SUBSCRIPTION_ID=SECRET_FILE` mapping
New version returns an unknown mutation outcome	Preserve the pinned route and reconcile before retry, rollback, or credential rotation
Old replica does not become offline	Keep new mutations paused; retain process and lifecycle evidence and follow the fleet recovery procedure

Retain deployment evidence

Retain one record containing:

- the source revision plus image and manifest digests;
- admission package and policy revisions;
- instance and replica identities, account ID, and public endpoint;
- host and peer DIDs, configuration and credential revisions, database

identity, and network policy;

- readiness response, lifecycle lease, first safe action, rollout decision,

drain result, and offline state.

This record proves only the observed deployment boundary. It does not prove external-product qualification, all-capability conformance, exactly-once provider effects, or a zero-downtime rollout.

Related documentation

- Deploy the connector fleet
- Connector host configuration
- Connector host lifecycle
- Cal.diy configuration
- [Cal.diy authentication and tenant

routing](../getting-started/authentication-and-tenant-routing.md)