Deploy the connector fleet

Use this guide to deploy one connector type, provider boundary, tenant-owned instance, and standalone host replica behind the product-neutral aipd gateway. It is for operators who control registry migration, signed admission, gateway and ho

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

The result is an admitted host with a current lifecycle lease and one bounded tenant capability binding. The procedure keeps the binding disabled until the host is ready, verifies tenant-scoped discovery before provider work, and starts with a non-mutating request.

This guide reflects source revision `97be86e9efedf07ecf1783b03800f683f107fb04`. It applies to standalone hosts for Cal.diy, Hermes Agent, Chatwoot, Dify, CrewAI, and Twenty. Deploy only the connector families and capabilities required by the reviewed deployment.

Use this procedure after production boundaries exist

Complete Deploy AIP in production first. You need:

- exact `aipctl`, `aipd`, connector control-plane, and host artifacts;
- an immutable digest for the selected host artifact;
- the manifest and implementation support map produced for that artifact;
- seven independently signed evidence documents and a signed admission

package;

- an admission trust policy with package and evidence-role roots;
- registry admin, gateway data-plane, and lifecycle database credentials with

different authority;

- core runtime and host runtime databases;
- gateway, lifecycle, and host signing identities;
- a tenant and provider account, workspace, application, or endpoint boundary;
- provider credentials in owner-controlled files;
- an external TLS path and private trust roots where applicable;
- approved non-mutating input and a rollback window.

If the connector has no standalone host artifact, follow Build a connector. If it still runs inside a bundled daemon, use Migrate bundled connectors to the fleet for the traffic transition.

This procedure does not create provider credentials, decide capability semantics, or qualify an external product. It deploys evidence and identities that already passed their owning review.

Freeze one fleet identity set

Assign stable identifiers before writing the registry:

Identity	Meaning	Change rule
Connector type	One implementation family	Stable across releases
Connector version	One immutable artifact, manifest, and support map	New artifact or contract requires a new version
Instance	One tenant-owned provider account or endpoint boundary	Account ownership cannot change in place
Replica	One concrete host process identity and endpoint	Pre-provision before host registration
Binding	One tenant, capability, and instance policy	Change through a higher policy revision
Package	One signed declarative catalog stream	Advance its monotonic revision for a new operation

Record the expected host endpoint, peer principal, DID, trust domain, topology, capacity, artifact digest, tenant membership, secret-provider reference, credential revision, and configuration revision. A runtime value that differs from its admitted record is a deployment failure, not a dynamic update.

Use one instance for one provider account boundary. Do not put several tenants behind a shared provider credential unless the connector contract and admission package explicitly define that ownership model.

Prepare separate database roles

Provision the connector registry with three authorities:

Role	Long-lived use	Authority
Registry administrator	None	Migrate schema, admit catalog packages, read the operator journal, and revoke packages
Gateway data plane	`aipd`	Read catalog and routes; reserve and settle assignments; maintain bounded fleet status
Lifecycle data plane	Connector control-plane service	Read admitted identity; register, heartbeat, drain, and mark replicas offline

Keep the administrator URL in a short-lived operator environment. Give the gateway and lifecycle service separate owner-controlled URL files. Reconcile the complete grant policy after every schema migration so stale permission is removed as well as new permission added.

Provision a separate core runtime database for all `aipd` replicas in this logical deployment. Provision a host-owned durable database for the selected connector boundary. Do not use the registry database as either runtime store.

Install the registry schema

Run migration from the exact `aipctl` artifact selected for the release:

```
SH
aipctl connector registry-migrate \
  --database-url-file /run/operator/registry-admin.url
```

Expected output is a JSON document with `status` set to `ok`, component set to `aip-connector-registry-postgres`, and the installed schema version. The command uses bounded control and verification pools, installs immutable migrations, and fails when the resulting version differs from the binary requirement.

After migration:

1. apply the reviewed gateway and lifecycle grants;
2. close the migration connection;

3. verify each long-running URL with the corresponding service startup check;
4. confirm neither URL can perform catalog administration;
5. retain the schema version, migration artifact, time, and operator identity.

Do not let a long-running process install a missing registry schema through administrator credentials. The gateway and lifecycle constructors require an already installed matching schema.

Prepare signed admission

The unsigned admission package binds one immutable deployment declaration:

- connector type and active version;
- bounded manifest and canonical manifest digest;
- artifact digest and connector SDK version requirement;
- implementation support for every callable capability;
- admission policy;
- tenant-owned instance;
- pre-provisioned offline replica;
- tenant capability bindings with `enabled` set to `false` for the first

package revision;

- OCI signature, SBOM, provenance, conformance, vulnerability-policy,

license-policy, and revocation-observation evidence.

Each evidence statement binds the same artifact and manifest digests, its own document digest, signer, policy outcome, issue time, and expiry. The trust policy supplies separate roots for the package and each evidence role.

The release authority signs the complete package after the seven signed evidence documents have been inserted:

```
SH
aipctl connector registry sign-package \
  --package admission-unsigned.json \
  --signing-seed-file /run/release/package-signing-seed.hex \
  --signer-identity connector-release > admission-signed.json
```

Keep evidence-role signing seeds outside this operator step. One release identity must not impersonate the build, scanner, conformance, license, or revocation authorities.

Plan and apply the disabled binding

Plan performs cryptographic and structural verification without writing:

```
SH
aipctl connector registry plan \
  --package admission-signed.json \
  --trust-policy admission-trust-policy.json
```

Review the returned package ID, revision, digest, connector type, version, artifact digest, instance count, replica count, binding count, and evidence count. Compare them with the release record, not with a mutable tag or filename.

Apply the exact verified bytes through short-lived administrator credentials:

```
SH
aipctl connector registry apply \
  --package admission-signed.json \
  --trust-policy admission-trust-policy.json \
```

```
  --database-url-file /run/operator/registry-admin.url
```

`apply` journals the package ID, revision, and digest. Repeating the same operation can resume safely. A different digest under the same package revision is a conflict.

Read the durable operator result:

```
SH
aipctl connector registry status \
  --package-id "$PACKAGE_ID" \
  --revision "$PACKAGE_REVISION" \
  --database-url-file /run/operator/registry-admin.url
```

Require the operation state to report a completed apply before starting the host. Close and unmount administrator credentials when the operator step ends.

Start the signed lifecycle service

Start one lifecycle service from its reviewed artifact. A loopback listener can share a pod or host with a TLS proxy:

```
SH
aip-connector-control-plane \
  --database-url-file /run/secrets/registry-lifecycle.url \
  --signing-seed-file /run/secrets/control-signing-seed.hex \
  --bind 127.0.0.1:8090
```

If the service binds to a non-loopback address, place it behind a trusted TLS proxy and pass the explicit proxy-network allowance. Do not expose its plaintext listener to an untrusted network.

Record the DID derived from the signing seed. Every host must pin this DID or load it from a regular public file. Verify:

- `/health` identifies the lifecycle component;
- `/ready` reports the installed schema version, catalog revision, and bounded

pool state;

- the retained control pool count is zero;
- `/metrics` is reachable only through the observability boundary;
- the database role cannot mutate connector types, versions, instances, or

bindings.

The lifecycle service exposes only signed register, heartbeat, drain, and offline commands. It cannot admit an unprovisioned connector identity.

Start the product-neutral gateway

Start `aipd` with core state, verified identity, and fleet data-plane access. This source-backed example shows the minimum fleet ownership classes; add only reviewed profile and limit settings:

```
SH
aipd \
  --bind "$AIPD_BIND" \
  --public-base-url "$AIP_PUBLIC_ORIGIN" \
  --service-id "$AIPD_PRINCIPAL_ID" \
  --trust-domain "$AIP_TRUST_DOMAIN" \
  --require-signed-envelopes \
  --trusted-signer-file /run/config/trusted-signers.json \
  --trusted-identity-file /run/config/trusted-identities.json \
  --approval-authority-file /run/config/approval-authorities.json \
  --postgres-url-file /run/secrets/aipd-runtime.url \
  --connector-registry-url-file /run/secrets/registry-data.url \
  --connector-fleet-signing-seed-file /run/secrets/gateway-signing-seed.hex \
  --connector-fleet-trust-registry-endpoints \
```

```
  --connector-fleet-tls-ca-file /run/config/fleet-ca.pem
```

Use an explicit host allowlist instead of registry-endpoint trust when that is the deployment policy. Plain HTTP and private-network exceptions are intended only for explicitly controlled environments and do not replace endpoint identity verification.

Require core `/ready` and inspect its fleet snapshot. Core readiness establishes storage, worker, required NATS, and required local-module state. It does not establish that this connector has a ready replica or an enabled tenant binding.

Configure one standalone host

Select the exact public host binary for the connector:

Connector	Host binary
Cal.diy	`aip-host-cal-diy`
Hermes Agent	`aip-host-hermes-agent`
Chatwoot	`aip-host-chatwoot`
Dify	`aip-host-dify`
CrewAI	`aip-host-crewai`
Twenty	`aip-host-twenty`

All six flatten the common host contract and add product-specific values. A deployment wrapper may supply the common contract through environment variables or flags. The common command shape is:

```
SH
"$HOST_BINARY" \
  --bind "$HOST_BIND" \
  --public-endpoint "$HOST_PUBLIC_ENDPOINT" \
  --control-plane-endpoint "$CONTROL_PLANE_ENDPOINT" \
  --control-plane-did-file /run/config/control-plane.did \
  --database-url-file /run/secrets/connector-runtime.url \
  --signing-seed-file /run/secrets/host-signing-seed.hex \
  --connector-type-id "$CONNECTOR_TYPE_ID" \
  --version-id "$CONNECTOR_VERSION_ID" \
  --instance-id "$CONNECTOR_INSTANCE_ID" \
  --replica-id "$CONNECTOR_REPLICA_ID" \
  --tenant-id "$TENANT_ID" \
  --membership-id "$MEMBERSHIP_ID" \
  --gateway-principal-id "$AIPD_PRINCIPAL_ID" \
  --gateway-did-file /run/config/gateway.did \
  --trust-domain "$AIP_TRUST_DOMAIN" \
  --artifact-digest "$HOST_ARTIFACT_DIGEST" \
  --secret-provider-ref "$SECRET_PROVIDER_REF" \
  "$@"
```

`HOST_PUBLIC_ENDPOINT` must end in `/aip/v1/messages` and resolve to the admitted host endpoint. `CONTROL_PLANE_ENDPOINT` must be the signed lifecycle route. The remaining arguments in `"$@"` are the reviewed product-specific configuration and secret-file references; do not pass arbitrary user input to that position.

Before start, compare every common identity with the disabled admission package. The host cannot repair a wrong type, version, instance, replica, tenant, endpoint, DID, artifact digest, or manifest at registration time.

Require registration and readiness

The common host startup sequence is:

1. validate identity, endpoints, limits, files, and trust;
2. discover the product connector manifest;

3. open the host PostgreSQL runtime;
4. recover durable runtime work;
5. register the pre-provisioned replica through the signed lifecycle service;
6. apply its lease and start heartbeat renewal;
7. serve process health, readiness, metrics, manifest, native AIP, and optional

product-owned ingress routes.

Do not enable the binding when only `/health` succeeds. Require all of:

- host `/ready` returns success;
- connector and storage probes are ready;
- the replica is not draining;
- its lease is current;
- the registry identity matches the intended artifact and endpoint;
- the running manifest digest matches admission;
- active assignments are zero before first traffic;
- gateway fleet maintenance reports no error.

If registration fails, keep the binding disabled. Do not create a replacement replica identity from the host process or weaken artifact, DID, tenant, or endpoint verification.

Enable one tenant binding

Prepare a second signed package revision from the same package stream. Preserve the immutable connector version, instance, replica, evidence, and artifact. Change only the intended binding to `enabled: true`, increase its policy revision, and advance the package revision.

Plan the second package while the host remains ready, then apply the exact planned bytes with the same short-lived operator sequence. Read its durable status and close the administrator credential again.

Configure `aipctl` with the approved principal, signing seed, trust domain, and expected gateway DID described in Use native AIP. Query the tenant-scoped catalog through that authenticated client:

```
SH
aipctl capability list "$AIP_PUBLIC_ORIGIN" \
  --capability-id "$CAPABILITY_ID" \
  --signed-native
```

The result must contain the expected capability definition for the authenticated tenant. Repeat with an unauthorized tenant and require that the binding is not visible. A catalog result proves tenant visibility; it does not prove provider execution.

Route the first safe action

Choose a capability whose reviewed contract is non-mutating and whose input cannot create a provider effect. Use the normal authenticated client boundary, not a direct host request:

```
SH
aipctl action call "$AIP_PUBLIC_ORIGIN" "$CAPABILITY_ID" \
  --input @safe-read.json
```

Require a terminal result consistent with the capability contract. Retain:

- action and correlation IDs;

- authenticated principal and tenant;
- capability and manifest digest;
- route assignment instance, replica, endpoint, peer DID, version, catalog

revision, binding revision, credential revision, health revision, and fence;

- host request and provider trace identifiers;
- result, events, receipts, and redacted audit records;
- gateway, host, and provider timestamps.

Confirm the request reached the intended provider account and that no bundled or second host path processed it. Only then admit a controlled mutation with a fresh idempotency key and the required approval or transaction context.

Scale without changing the contract

Add capacity in separate changes:

1. pre-provision a new replica identity and endpoint in a signed package;
2. plan and apply the package revision;
3. start the exact admitted artifact;
4. require registration, current lease, readiness, and zero assignments;
5. observe routing at the intended region, zone, and capacity class;
6. expand capacity or tenant binding policy only after evidence is retained.

Do not reuse a replica ID across two live processes. Do not change artifact, manifest, endpoint, peer identity, tenant, or topology under an existing replica identity.

Existing actions retain their pinned route for retry, cancellation, and reconciliation. New ready capacity does not authorize moving an uncertain provider operation to another replica.

Resolve deployment failures

Symptom	Decision
Registry migration reports a checksum or version mismatch	Stop; use the reviewed migration source and inspect the retained schema record
Admission plan fails	Correct signatures, evidence, time bounds, digests, SDK range, manifest, or registry relationships; do not apply
Apply is interrupted	Read package status and resume the same revision and digest
Lifecycle readiness fails	Inspect schema version, lifecycle grants, pool bounds, and database connectivity
Gateway rejects fleet configuration	Supply the complete registry, signing, endpoint-trust, and positive-bound set
Host admission fails	Compare type, version, instance, replica, tenant, membership, endpoint, DID, digest, and manifest
Host health passes but readiness fails	Inspect runtime storage, connector probe, lease, heartbeat, and drain state
Catalog omits the capability	Inspect authenticated tenant, enabled binding, active version, implementation support, and catalog revision
No ready replica is available	Keep mutations off; inspect lease, status, capacity, topology, and circuit state
First action has an unknown outcome	Reconcile it through the pinned route; do not retry through another host

Preserve admission journals, route assignments, provider operation IDs, idempotency records, lifecycle transitions, and audit evidence while diagnosing.

Drain and roll back

Prepare a signed rollback package revision with the affected binding disabled and a higher binding policy revision. For a normal rollback:

1. pause new mutations;
2. plan and apply the disabled-binding package;
3. verify tenant discovery no longer exposes that route;
4. begin host drain through its normal shutdown path;
5. wait for active assignments, callbacks, events, and provider jobs to settle;
6. reconcile every unknown outcome through its pinned replica;
7. require the lifecycle transition to `offline`;
8. stop the host and retain its durable database;
9. restore the previous binding or bundled path only after overlap is excluded.

If the artifact or its evidence is compromised, revoke the applied package:

```
SH
aipctl connector registry revoke \
  --package-id "$PACKAGE_ID" \
  --revision "$PACKAGE_REVISION" \
  --reason "$REVOCATION_REASON" \
  --database-url-file /run/operator/registry-admin.url
```

Revocation marks the exact connector version revoked and stops new traffic for it. It does not terminate provider work, undo external effects, delete audit state, or revoke a provider credential. Perform those actions through their own reviewed boundaries.

Accept the fleet boundary

Accept this deployment only when the record contains:

- exact component and host artifact digests;
- registry schema and grant revisions;
- package ID, revision, digest, signer, seven evidence statements, and status;
- type, version, instance, replica, binding, tenant, account, endpoint, DID,

topology, capacity, and credential revisions;

- lifecycle registration, readiness, heartbeat, drain, and offline evidence;
- authorized and unauthorized discovery results;
- first read and controlled mutation traces where applicable;
- recovery, restart, scale, and rollback results;
- known exclusions, evidence expirations, and claim owner.

Admission proves that an exact catalog package satisfied the selected trust policy. Registration proves that an exact host acquired a lease. Discovery proves tenant visibility. An action proves only its observed path. Report qualification and production readiness only when their broader evidence campaigns also pass.

Related documentation

- Capabilities and contracts
- Actions and sessions
- Connector catalog