Deploy the Dify connector

Use this guide to deploy one immutable aip-host-dify instance. The result is an admitted, observable route with durable cancellation evidence and a defined replacement and rollback path.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/dify/operations/deployment.md

The repository fleet stack is qualification evidence. It is not a production deployment template.

Define the deployment boundary

One logical instance owns:

- one admitted connector type and immutable version;
- one tenant and membership;
- one fixed Dify origin;
- a reviewed set of app and Knowledge descriptors;
- one scoped key source per provider credential;
- one coherent durable host-state boundary;
- one or more uniquely identified replicas;
- a fixed native AIP endpoint per replica.

The process serves native AIP on `0.0.0.0:8081` by default. Its advertised endpoint must be an absolute HTTPS URL ending in `/aip/v1/messages`.

Keep the native port behind the selected TLS edge. Only explicit loopback development may opt into HTTP.

Map topology ownership

Connection	Direction	Owner	Required protection
Gateway to Dify host	Inbound native AIP	AIP platform	TLS, signed envelopes, trusted gateway identity
Lifecycle service to host	Inbound control	AIP platform	Fixed endpoint, DID verification, lease fencing
Host to Dify	Outbound provider API	Connector operator	HTTPS origin, scoped bearer keys, no authenticated redirects
Host to PostgreSQL	Durable runtime state	Connector operator	Dedicated URL file, migrations, backups
Host to stream callback	Optional outbound publication	AIP platform	Fixed endpoint and explicit network policy

Replicas of one logical instance share the admitted version and coherent durable state. Each replica retains its own replica ID and signing identity.

1. Freeze the release unit

Record these values before provisioning:

Release evidence	Required value
AIP source revision	Exact commit
Host image	Immutable OCI digest
Workspace version	Version embedded in the artifact
Connector type and version	Exact admitted identities
Dify compatibility baseline	Exact reviewed provider revision
Descriptor set	Hashes of public app and Knowledge files
Manifest	Exact projected capability set and digest
Configuration revision	Positive monotonic value
Admission package	Signature bound to artifact and manifest
Trust policy	Exact policy used for admission

Do not deploy from a mutable image tag alone. A descriptor ID or mode changes the manifest, so it requires a new immutable version and admission package.

The source fleet preparation script records a source digest, image ID, release ID, and release revision. Apply the same identity discipline to the actual release pipeline.

2. Provision durable state

Place the owner-only PostgreSQL URL in the file referenced by:

```
TEXT
AIP_CONNECTOR_HOST_DATABASE_URL_FILE
```

The prepared host uses durable state for Action handling, lifecycle recovery, profile state, replay fencing, and stream publication. The Dify binary also installs `ProfileStateDifyTaskStore` after host preparation.

That task store maps AIP actions to remote task IDs under the logical instance scope. It preserves cancellation evidence across restart and replica handoff.

All replicas for one instance need the same storage authority. Process-local task state is not a production substitute.

Migrate and verify storage before a replica registers. Back up runtime and profile state as one documented recovery unit.

3. Place identity and trust material

Provision the shared connector-host identity:

- Dify connector type and immutable version IDs;
- logical instance and unique replica IDs;
- tenant and membership IDs;
- artifact digest and public native endpoint;
- trust domain, gateway identity, and control-plane identity;
- replica signing seed file;

- optional private-PKI root;
- complete credential revision and policy fields when credential fencing is

enabled.

Signing seeds, database URLs, Dify keys, and private CA material belong in owner-controlled files. Environment variables carry file paths and non-secret identifiers.

The public endpoint, manifest, artifact, version, and signer must agree with the admitted record.

4. Configure the provider boundary

Set one fixed Dify origin and at least one descriptor domain:

```
TEXT
AIP_DIFY_BASE_URL=https://dify.example.test
AIP_DIFY_APPS_FILE=/run/config/dify-apps.json
AIP_DIFY_KNOWLEDGE_FILE=/run/config/dify-knowledge.json
```

The app file can use one owner-only key file per app. A legacy global app key is permitted only when all configured apps intentionally share it.

Every Knowledge descriptor requires its own workspace-scoped key file. The global app key never supplies Knowledge authorization.

Keep key material out of descriptor files. Mount public descriptors read-only and secret files through a separate bounded secret path.

An optional external-account identity must include both ID and system. Define its deployment meaning because one host can contain several app and Knowledge identities.

5. Check manifest capacity

Compute the exact manifest before admission:

```
TEXT
sum(per-app projected capabilities) + 46 × Knowledge credential count
```

Descriptor parser ceilings allow up to 1,000 apps and 1,000 Knowledge credentials. Global manifest admission normally establishes a lower practical limit.

Split logical instances before weakening admission bounds. Preserve unique descriptor IDs and explicit tenant ownership across the split.

6. Admit before starting

Apply the connector admission package and trust policy before the replica starts. Admission must bind:

- connector type and immutable version;
- semantic manifest and capability identities;
- artifact digest and provenance;
- native and Dify profile support;
- tenant and external-account policy;
- public endpoint and lifecycle expectations.

Use separate registry credentials for admission, lifecycle control, and data-plane reads. The host does not need an administrator credential.

The reviewed fleet graph completes `admit-dify` before `dify-acme-a`. Preserve that dependency even when production service names differ.

7. Start dependencies in order

Use this order:

1. PostgreSQL and required migrations;
2. connector registry and the admitted Dify version;
3. control plane and trusted gateway;
4. TLS edge and fixed internal routes;
5. optional stream callback ingress;
6. Dify provider reachability and credential files;
7. the Dify host replica;
8. lifecycle registration and lease renewal;
9. readiness and safe canaries;
10. production route enablement.

Stop when an earlier authority is unavailable. Never start with a temporary identity, unadmitted version, mutable digest, or substitute provider origin.

8. Gate readiness

Require all applicable signals:

Gate	Established fact
Process	Host server and local supervisor are running
Durable store	Runtime and profile-state operations are available
Admission	Type, version, instance, replica, artifact, and manifest match
Lifecycle	Replica is registered, healthy, leased, and not draining
App health	Each app key reads `/v1/parameters`
Knowledge health	Each Knowledge key reads `/v1/datasets?limit=1`
Route	Gateway resolves the admitted instance and replica
Callback, when configured	Endpoint and network policy are valid

Product health proves credential and route reachability at one moment. It does not qualify every projected operation or verify the configured app mode.

9. Run read-only canaries

Before admitting mutations:

1. read the discovered manifest through the tenant route;
2. compare exact IDs and expected capability counts;
3. invoke one app `parameters.get` when apps are configured;
4. invoke `dataset.list` when Knowledge is configured;
5. retain Action, gateway, replica, and provider request identities;

6. verify the intended tenant, descriptor, and key served each action.

Do not use model execution, upload, indexing, deletion, training, or pipeline run as the first deployment canary.

10. Enable traffic gradually

Start with a bounded tenant and capability subset. Observe admission, completion, provider authentication, response bounds, task-store health, cancellation certainty, and lifecycle leases.

Enable mutations only after approval and idempotency paths are exercised with controlled data. Enable streaming after callback ownership and durable task correlation are proven.

Readiness is necessary but does not establish rollout success.

Replace a replica

For a same-version replacement:

1. provision a new replica ID and signer;
2. use the same logical instance and coherent durable state;
3. start it with byte-identical product configuration;
4. wait for registration, lease, health, route, and canary gates;
5. stop new assignments to the old replica;
6. let pinned work settle within the drain boundary;
7. mark the old replica offline and stop it;
8. retain lifecycle and Action evidence.

Product configuration is startup-owned. A key, descriptor, origin, response limit, or manifest change requires replacement rather than live mutation.

Roll back safely

Keep the previous artifact, admission record, descriptors, credential revision, and compatibility decision until the new version clears its rollout window.

Rollback routes new work to a previously admitted compatible version. It does not overwrite the failed version in place.

Preserve durable Action state, idempotency records, task correlation, provider request identity, and pinned work. Never replay an ambiguous provider mutation under a new Action ID.

Retain deployment evidence

Retain:

- source revision, image digest, SBOM, and provenance identity;
- connector type, version, instance, replica, and tenant;
- descriptor, manifest, and configuration hashes;
- admission package and trust-policy identity;
- credential revision without secret contents;
- migration, startup, lease, health, route, and canary outcomes;

- rollout, drain, replacement, rollback, and operator decisions.

Redact provider payloads and never store key contents as evidence.

Completion criteria

Deployment is complete only when:

- immutable artifact and manifest identities match admission;
- descriptors and keys have explicit least-privilege ownership;
- durable runtime and task correlation survive replica replacement;
- identity, control, gateway, TLS, and provider dependencies are healthy;
- the replica holds a valid non-draining lease;
- exact manifest and read-only canaries pass;
- streaming has callback and cancellation ownership when enabled;
- rollback material and procedures remain available;
- no mutation substitutes for a safe deployment check.

These gates establish a controlled deployment. They do not qualify every Dify capability against a production provider.

Related documentation

- Dify connector configuration
- Run the Dify connector quickstart
- Production deployment
- Deploy the connector fleet
- Connector host lifecycle