Deploy the CrewAI connector

Use this procedure to deploy one CrewAI connector instance as two separately identified processes. The Rust host owns AIP identity and lifecycle. The Python sidecar owns CrewAI execution and its durable job journal.

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

The reviewed Compose file is a qualification topology, not a production manifest. Adapt its security and dependency boundaries to the target platform without copying qualification identities or placeholder digests.

Fix the deployment unit

One logical connector instance contains these components:

Component	Reviewed artifact	Listener	Durable ownership
Rust host	`getaip/aip-host-crewai`	Native AIP on `8081` behind ingress	PostgreSQL runtime store through an owner-controlled URL file
Python sidecar	`getaip/aip-crewai-sidecar`	Private HTTP on `8090`	CrewAI home, training directory, and `runs.json` on one dedicated volume
Connector registry	Admitted CrewAI type, version, instance, and replica	Control-plane boundary	Admission package, assignments, leases, and lifecycle state
Private ingress	Deployment-owned TLS route	Host and sidecar service names	Trust roots and routing configuration

Do not combine the processes into one artifact identity. Their source inputs, dependencies, image digests, credentials, and state have different review and rollback boundaries.

1. Freeze both artifacts

Retain an immutable digest, provenance, software bill of materials, source revision, and vulnerability decision for each image.

The reviewed generic host Dockerfile builds `aip-host-crewai` with locked Cargo dependencies. Its runtime image uses UID and GID `10001`, exposes `8081`, and contains the common binary as `aip-connector-host`.

The default sidecar Dockerfile uses Python 3.13 and a frozen `uv` lockfile. It labels the CrewAI package set as `1.15.5`, runs as UID and GID `10001`, and exposes `8090`.

Reject mutable tags as deployment identity. OCI source and revision labels are useful metadata but do not prove the bytes behind a registry digest.

2. Admit the host artifact

Create or verify the immutable connector type and version before starting the replica. Bind the admitted version to the exact host artifact digest and required evidence package.

Provision unique values for:

- connector instance ID;
- connector replica ID;
- tenant and membership IDs;
- external CrewAI account reference;
- artifact digest and configuration revision;
- region, zone, capacity class, and trust domain.

Each replacement artifact requires a version and assignment decision. Do not reuse a replica ID for two simultaneously running processes.

3. Provision owner-controlled files

Mount these host inputs as read-only files:

Input	Purpose
PostgreSQL URL	Durable AIP runtime and replay state
Ed25519 signing seed	Native AIP response and lifecycle identity
Sidecar Bearer token	Private host-to-sidecar authentication
Crew descriptor array	Admitted crews and exact operation allowlists
Credential revision policy	Reloadable fail-closed provider authority
Gateway and control-plane DIDs	Verification of signed fleet traffic
Private CA certificate	TLS validation for private deployment routes

Use separate credentials for PostgreSQL, AIP signing, sidecar authentication, and any model or tool provider. Rotation of one authority must not silently replace another.

The host accepts one to 1,000 crew descriptors and a sidecar token file up to 16 KiB. File readers enforce bounded, owner-controlled inputs; preserve those controls in secret projection and rotation workflows.

4. Prepare the sidecar state

Create one protected writable volume for:

```
TEXT
/var/lib/aip-crewai/
■■■ runs.json
■■■ runs.json.lock
■■■ training/
■■■ crewai/
■■■ home/
```

The journal and lock file appear after startup. The default image places CrewAI storage and `HOME` under the same volume, preserving its local credential-encryption material across restarts.

Give the sidecar process ownership of the directory. Do not mount the same writable state path into a second active sidecar. The nonblocking journal lock is a startup fence, not a shared-volume scaling mechanism.

5. Isolate the sidecar route

Place the sidecar on a private network reachable by its paired Rust host and the deployment health system. Do not publish port `8090` to tenants or the public AIP edge.

Terminate or route private TLS so the host can use an `https` sidecar URL. The Rust connector rejects non-loopback HTTP and disables redirects.

The sidecar itself provides Bearer authentication, not TLS or AIP tenant identity. Configure the private ingress to preserve the fixed route and avoid forwarding untrusted proxy identity.

6. Start and verify the sidecar

Configure at least:

- registry module reference;
- Bearer token file;
- state file;
- allowed startable operations;
- concurrency and journal bounds;
- training directory when `train` is enabled.

Keep the container filesystem read-only, drop Linux capabilities, enable `no-new-privileges`, and provide a bounded no-execute temporary filesystem.

The reviewed healthcheck reads the Bearer token file and calls authenticated `GET /health` on loopback. Require HTTP `200`, the expected crew IDs, durable journal status, enabled operations, and expected capacity values.

Health at this point proves sidecar startup and local configuration. It does not prove model credentials, tool reachability, or successful crew execution.

7. Start the Rust host

Start the host only after:

1. the connector version is admitted;
2. PostgreSQL and the lifecycle control plane are ready;
3. gateway and private TLS trust roots are available;
4. the sidecar passes authenticated health;
5. all required files and immutable identities are mounted.

Set the sidecar URL, sidecar token file, and crew descriptor file together with the common connector-host configuration. The reviewed host binds `0.0.0.0:8081` behind ingress and publishes a distinct HTTPS AIP endpoint.

The host discovers capabilities from the descriptor array during preparation. Configuration or discovery failure prevents registration and readiness.

8. Gate traffic on host readiness

Probe `/ready` for routing eligibility. Keep `/health` for process health and diagnosis. The reviewed qualification topology routes only after dependencies and the host readiness probe pass.

Verify the registered type, version, instance, replica, artifact digest, tenant, external account, zone, credential revision, lease, and endpoint. Compare discovered CrewAI capabilities with the admitted manifest.

Send no tenant traffic until the registry assignment targets this exact ready replica. Sidecar health alone is insufficient for fleet routing.

9. Scale with independent instances

Do not add a second sidecar replica against the first sidecar's journal. Instead, create another connector instance with:

- its own instance and replica identities;
- its own host runtime state and signing seed;
- an independent sidecar state volume and token;
- an explicit crew and provider-account assignment;
- its own admitted route weight and lifecycle state.

This model prevents two Python processes from claiming one durable job namespace. Fleet routing supplies horizontal capacity above the instance boundary.

10. Replace or roll back safely

Drain the host through the lifecycle control plane before removal. Stop new assignments, preserve runtime and sidecar state, and wait only within the configured drain deadline.

Reconcile every non-terminal or uncertain Action before replacing its sidecar state. A new empty volume removes the old request fence and can permit duplicate provider work under a newly dispatched Action.

For rollback, deploy the previously admitted immutable host and sidecar pair with compatible configuration. Use a new replica identity, verify both health boundaries, then move assignments deliberately.

Retain the retired image digests, journal, events, host checkpoints, registry history, and operator decision until the replay and incident-retention windows close.

Verify the deployment

Accept the deployment only when:

- both running image digests match retained provenance;
- the sidecar route is private and authenticated;
- exactly one process owns each sidecar journal;
- the expected crews and operation policy appear in health;
- the Rust host reports healthy and ready separately;
- registry identity matches the running replica and artifact;
- the host advertises only descriptor-approved capabilities;
- one bounded qualification Action completes with retained evidence;
- drain and rollback procedures preserve the original Action fence.

Related documentation

- CrewAI connector overview
- CrewAI connector configuration
- CrewAI sidecar contract and lockfile
- CrewAI idempotency, cancellation, and process fencing
- Deploy the connector fleet