Deploy the Twenty connector

Use this procedure to deploy one standalone aip-host-twenty instance. The host owns AIP identity, fixed provider workspace access, durable Action state, optional webhook replay, local events, and central publication.

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

The reviewed connector-fleet Compose file does not contain a Twenty service. Use the source-owned binary and generic host image as inputs to a separately reviewed deployment rather than treating that Compose topology as a template.

Fix the deployment unit

One logical instance requires:

Component	Role	Durable ownership
`aip-host-twenty`	AIP endpoint, provider mapping, health, webhook ingress	Connector runtime PostgreSQL
Twenty deployment	Fixed record and metadata provider	Provider-owned workspace state
Connector control plane	Type, version, instance, replica, lease, lifecycle	Registry and admission state
AIP gateway	Signed tenant traffic and route selection	Gateway runtime state
Private ingress	Host AIP, lifecycle, and optional webhook routes	TLS and routing configuration
Secret manager	Database URL, signing seed, API token, optional webhook secret	Versioned secret material

The Twenty provider does not run inside the connector host. Keep its lifecycle, backup, schema, and credential ownership separate from the AIP artifact.

1. Freeze the host artifact

The generic host Dockerfile accepts this exact package and binary pair:

```
TEXT
PACKAGE=aip-host-twenty
BINARY=aip-host-twenty
```

A representative source build is:

```
SH
docker build \
  --file deploy/connector-fleet/Dockerfile.host \
  --build-arg PACKAGE=aip-host-twenty \
  --build-arg BINARY=aip-host-twenty \
  --build-arg SOURCE_REVISION=97be86e9efedf07ecf1783b03800f683f107fb04 \
  --build-arg IMAGE_VERSION=1.0.0 \
  --tag getaip/aip-host-twenty:reviewed \
  .
```

Run this command only from a clean checkout whose exact revision and lockfiles have passed the deployment's build policy. The mutable example tag is not an artifact identity.

The runtime image:

- uses a pinned Alpine base in source;
- runs as UID and GID `10001`;
- exposes port `8081`;
- installs the binary as `/usr/local/bin/aip-connector-host`;
- uses that binary as its entrypoint;
- checks loopback `/health` with `wget`.

Retain immutable OCI digest, source revision, Cargo lock identity, provenance, software bill of materials, vulnerability decision, labels, and build logs.

The source-owned product image verifier includes `aip-host-twenty`, checks its help output, non-root user, entrypoint, component label, revision, and version. That build check is input to deployment review, not runtime qualification.

2. Admit the artifact and instance

Create or verify an immutable connector type and version before starting the replica. Bind the version to the exact image digest and required evidence.

Provision unique values for:

- connector instance ID;
- connector replica ID;
- tenant and membership IDs;
- external-account ID and system;
- artifact digest and configuration revision;
- region, zone, capacity class, and trust domain;
- provider workspace UUID and credential revision.

For a tenant-owned workspace, deployment policy should require external-account ID to equal the configured workspace UUID. The host does not enforce that equality automatically.

Do not reuse a replica ID for two running processes. Do not change the provider workspace behind an unchanged logical instance without a new admission and configuration decision.

3. Provision owner-controlled files

Mount these required host inputs:

Input	Secret	Purpose
PostgreSQL URL file	Yes	Durable runtime, replay, event, and outbox stores
Ed25519 signing-seed file	Yes	Host response, lifecycle, and callback identity
Twenty API-token file	Yes	Fixed provider Bearer authentication
Gateway DID file or value	No	Verify signed gateway traffic
Control-plane DID file or value	No	Verify lifecycle responses
Private CA file	No	Validate private deployment TLS when used

Input	Secret	Purpose
Operation allowlist file	No	Non-empty subset of 22 fixed suffixes
Credential-revision policy file	No	Reloadable fail-closed authority fence

The database URL and API token have 16 KiB file bounds. The signing seed has a 256-byte file bound and must encode exactly 32 bytes as hexadecimal. The operation file has a 128 KiB bound.

Secret files must be regular, non-symlink, owner-controlled inputs with no group or world permission bits on Unix. Mount them read-only to the host.

Provision the optional webhook HMAC secret as a separate owner-controlled file. Never reuse the API token, signing seed, database credential, or caller token as the webhook secret.

4. Prepare durable PostgreSQL

The standalone host uses PostgreSQL for shared runtime state. One logical instance needs durable access across replica restart and replacement.

Preserve at least:

- Action and idempotency records;
- execution checkpoints and receipts;
- connector profile state;
- webhook replay nonces when ingress is enabled;
- local event records;
- durable connector-event outbox batches;
- registry lease and lifecycle evidence owned by their services.

Apply database TLS, least-privilege role, encryption, backup, restore testing, monitoring, and retention policy outside the connector.

Do not deploy a webhook-enabled second replica with an independent replay database under the same instance identity. Every replica of one instance must share the profile-state replay scope.

5. Isolate network routes

Allow the host to reach only:

- the fixed Twenty origin;
- its PostgreSQL service;
- the lifecycle control plane;
- the trusted AIP gateway path;
- the configured central event endpoint when webhooks are enabled;
- required private-PKI services according to deployment policy.

Expose the native AIP endpoint only through authenticated ingress. The public endpoint supplied to the host must end in `/aip/v1/messages`.

The provider origin must be HTTPS unless controlled loopback or explicitly allowed private HTTP applies. It contains no credential, path, query, or fragment, and provider redirects remain disabled.

If webhooks are enabled, expose only the deployment-owned external route that maps to internal `POST /webhooks/twenty`. Preserve all required headers and the raw body exactly.

6. Configure the provider binding

Set the product-specific values together:

```
TEXT
AIP_TWENTY_BASE_URL=https://crm.example.test
AIP_TWENTY_WORKSPACE_ID=0190c42f-2d5a-7000-8000-000000000090
AIP_TWENTY_API_TOKEN_FILE=/run/secrets/twenty-api-token
AIP_TWENTY_OPERATIONS_FILE=/run/config/twenty-operations.json
AIP_TWENTY_MAX_RESPONSE_BYTES=67108864
```

Omit the operations file only when all 22 source operations are intentionally admitted. Keep critical metadata and permanent-destroy operations disabled until policy, procedures, and evidence are ready.

Configure all common identity, database, signing, gateway, control-plane, topology, artifact, secret-provider, lifecycle, and optional credential-handle values from one reviewed deployment record.

7. Add webhook ingress only when required

For a webhook-enabled instance, add:

```
TEXT
AIP_TWENTY_WEBHOOK_SECRET_FILE=/run/secrets/twenty-webhook-hmac
AIP_CONNECTOR_HOST_EVENT_ENDPOINT=https://aipd.example.test/aip/v1/events
```

Configure event destination TLS and private-network exceptions through the shared host settings. An invalid or unauthorized event route fails startup.

The host derives replay scope from `instance:`, mounts the webhook route, and declares channel `twenty-webhooks`. Omission of the secret leaves the route unmounted.

Provider webhook metadata is a separate critical mutation. Create it after the host ingress, replay store, local event log, and central outbox are ready.

8. Start in dependency order

Start the host only after:

1. the immutable connector version is admitted;
2. PostgreSQL is reachable through its protected URL file;
3. lifecycle control plane and gateway trust identities are available;
4. private TLS roots and fixed provider routing are ready;
5. all required secret and configuration files are mounted;
6. the expected provider workspace and least-privilege token are active;
7. optional event ingress and outbox destination are authorized;
8. the target instance and replica assignments are provisioned.

During preparation, the host validates configuration, opens durable storage, constructs signed identity, and discovers its manifest before registration. Failure prevents normal registration and readiness.

Do not bypass an invalid operation suffix, workspace UUID, origin, secret file, DID, credential revision, or manifest.

9. Gate traffic on health and readiness

Use `/health` for process-level diagnosis. Use `/ready` for route eligibility. The generic image healthcheck probes only loopback `/health`.

The connector-specific health check sends authenticated `GET /rest/open-api/core` to the fixed provider origin. A successful HTTP status proves path reachability for that credential at that moment.

Before tenant traffic, verify:

- running image digest equals the admitted artifact;
- type, version, instance, replica, tenant, external account, and topology match;
- credential and configuration revisions match the deployment record;
- PostgreSQL, lifecycle lease, host health, and host readiness pass;
- discovered capabilities equal the intended operation subset;
- unauthorized tenants have no eligible route;
- a bounded read returns data from the intended workspace;
- optional webhook route and central event path pass a controlled test.

Health does not validate every operation, mutation permission, schema, replay race, or event recovery path.

10. Scale with shared instance state

Multiple replicas of one logical instance need:

- unique replica IDs and signing identity decisions;
- the same admitted connector version and compatible configuration;
- shared durable PostgreSQL for Action and profile state;
- the same provider workspace and credential policy;
- one coherent operation allowlist and artifact identity;
- one instance-scoped webhook replay boundary when ingress is enabled;
- explicit route weights, zones, capacity, leases, and drain ownership.

Do not scale by copying one replica ID, creating an independent replay database, or changing workspace ownership on one replica.

Run multi-replica qualification before claiming cross-replica idempotency, webhook replay, event outbox recovery, or failover behavior for the artifact.

11. Replace or roll back safely

Drain the old replica through the lifecycle control plane before removal. Stop new assignments, preserve durable state, and observe in-flight Actions within the configured drain deadline.

Before replacement, classify every nonterminal or uncertain mutation. Preserve the original Action, key, provider request identity, checkpoint, result, and receipts. Do not replay it to prove the replacement works.

Start a replacement with a new replica ID, the intended immutable image, the same logical instance state, and a reviewed configuration revision. Verify health, readiness, registry identity, capability subset, and workspace read before moving assignments.

For rollback, use a previously admitted immutable artifact whose schema, database, operation allowlist, and provider mapping remain compatible. Keep the new replica available only within the explicit rollback window.

Retain retired identities, image digests, registry history, database snapshots, webhook replay and outbox state, checks, drain evidence, and operator decision until incident and replay-retention obligations close.

Verify the deployment

Accept deployment state only when:

- immutable host digest and provenance match admission;
- required files pass ownership, type, size, and secret-custody checks;
- fixed provider origin, workspace, external account, and credential align;
- PostgreSQL backup and restore responsibilities are assigned;
- network policy permits only intended dependencies and ingress;
- process health and route readiness are independently verified;
- registry identity and discovered operation subset match the running replica;
- unauthorized tenant and credential-revision checks fail closed;
- one bounded read completes with retained evidence;
- optional webhook delivery is accepted, persisted, and centrally observed;
- drain, replacement, and rollback preserve original Action and replay state.

This checklist accepts a deployment configuration. Connector qualification still requires an exact procedure, artifact, provider identity, results, and limitations.

Related documentation

- Twenty connector overview
- Twenty connector configuration
- Authentication and workspace isolation
- Connector host lifecycle
- Deploy the connector fleet