Deploy AIP in production

Use this guide to design and stage a production AIP deployment. It is for operators who own the authenticated edge, product-neutral getaip-server replicas, durable state, connector registry, lifecycle control plane, standalone connector hos

SECTION	Deploy and Operate
TYPE	Guide
PROTOCOL	AIP 1.0
SOURCE	docs/guides/production-deployment.md

The result is a bounded deployment candidate with explicit ownership, trust, traffic gates, recovery, and rollback. Production readiness remains a claim about an exact artifact set, configuration, topology, provider boundary, time, and retained evidence. A source revision, successful build, health response, or admission package does not establish that claim by itself.

This guide reflects source revision `d7cce13d1d555644d04a4d73c66c95b113737635`. It does not prescribe a cloud, orchestrator, ingress controller, PostgreSQL service, NATS service, or secret manager.

Use this procedure for a complete deployment boundary

Apply this procedure after the selected AIP artifacts and connector contracts have been reviewed. Complete Install AIP before planning the production environment.

You need authority to:

- approve exact gateway, control-plane, host, and migration artifacts;
- provision networks, identities, databases, keys, and secret references;
- apply registry migrations, grants, admission packages, and tenant bindings;
- change external routes, provider ingress, and client traffic;
- take and restore backups;
- pause mutations, drain workloads, and execute rollback;
- retain qualification and deployment evidence.

Stop before deployment if the team cannot name one owner for identity, database recovery, connector admission, provider credentials, ingress, incident response, and rollback. Shared infrastructure does not remove these ownership boundaries.

Use the product-neutral topology

The production boundary separates client-facing protocol handling from product execution. `getaip-server` authenticates requests, owns AIP lifecycle state, discovers tenant-visible capabilities, and routes connector work. Standalone connector hosts own provider configuration, credentials, product calls, and authenticated product ingress.

```
MERMAID
flowchart LR
    Clients["Native AIP, MCP, and A2A clients"] --> Edge["TLS and authenticated edge"]
```

```
Edge --> Core["Product-neutral getaip-server replicas"]
Core --> RuntimeDB[("Shared AIP runtime PostgreSQL")]
Core -. optional .-> NATS["Authenticated NATS fabric"]
Core --> Registry[("Connector registry data plane")]
Identity["Identity and approval authorities"] --> Core

Release["Release and admission operator"] --> RegistryAdmin["Registry migration and admission"]
RegistryAdmin --> Registry
Control["Signed connector lifecycle control plane"] --> Registry

Core --> Hosts["Admitted standalone connector hosts"]
Hosts --> Control
Hosts --> HostDB[("Host-owned durable state")]
Hosts --> Products["Product APIs"]
Products --> Ingress["Authenticated product ingress"]
Ingress --> Hosts
Hosts --> Events["Signed central event and callback ingress"]
Events --> Core
```

Keep these roles distinct even when one managed PostgreSQL cluster or one orchestrator runs several components. Do not embed product connector code or provider credentials into the product-neutral `getaip-server` artifact.

The repository contains a connector-fleet Compose topology for controlled development and qualification. Its presence is evidence of a reference path, not a production deployment template or a claim about any external provider.

Define the release unit and claim

Freeze one deployment release record before provisioning. Include:

Release element	Identity to retain
AIP source	Full commit and source-tree digest
Gateway	Immutable artifact digest and AIP schema set
Registry migration	Exact binary or migration source and required schema version
Lifecycle control plane	Immutable artifact digest and signer DID
Connector host	Product, source revision, immutable digest, and manifest digest
Admission	Signed package ID, revision, digest, trust policy, and evidence expirations
Configuration	Redacted revision and deployment-owned defaults
Trust	Principal IDs, DIDs, trust domain, issuer, audience, and authority revisions
Storage	Database identities, schema revisions, backup IDs, and restore procedure
Provider	Account, workspace, application, endpoint set, and upstream revision
Qualification	Exact topology, procedure, result, time, and retained artifacts

The release unit may contain only the connector families the deployment needs. The public source provides standalone hosts for Cal.diy, Hermes Agent, Chatwoot, Dify, CrewAI, and Twenty, but evidence for one does not qualify the others.

Define the intended production claim in one sentence. Name the tenants, profiles, capabilities, connector versions, provider accounts, topology, and evidence window. Anything outside that sentence remains unsupported by this deployment record.

Segment networks and authority

Create separate network and credential boundaries for:

Zone	Permitted communication
Public edge	Client TLS to the published AIP, MCP, or A2A origin
Core service	Edge to `getaip-server`; `getaip-server` to runtime storage, registry data plane, and admitted hosts
Registry administration	Short-lived migration and admission jobs to registry control credentials
Lifecycle	Connector hosts to the signed lifecycle endpoint and lifecycle database role
Connector host	Gateway and lifecycle traffic, host database, approved product endpoints, and central ingress
Product ingress	Provider delivery to the product-specific authenticated host route
Observability	Read-only collection of bounded health, readiness, metrics, logs, and traces

The connector registry source supports independently permissioned control and data pools. Production gateway and lifecycle processes connect through the data-plane constructor, which verifies an already installed schema and retains no registry control pool. The reference grant policy limits gateway routing authority and lifecycle replica authority to different table operations.

Keep catalog migration and admission credentials in short-lived operator jobs. Do not mount them into `getaip-server`, the lifecycle service, or connector hosts.

Terminate external TLS before the core listener and keep the listener on a protected service network. A non-loopback `getaip-server` listener requires an external HTTPS origin. That origin must contain no credentials, path, query, or fragment. Plain HTTP is accepted only for explicit loopback development.

Provision durable state by owner

Treat runtime, registry, and connector-host state as separate recovery domains:

State owner	Durable content	Access pattern
Core runtime	Actions, results, sessions, events, approvals, transactions, callbacks, receipts, replay, and idempotency	Shared by all replicas in one logical `getaip-server` deployment
Connector registry	Types, versions, manifests, instances, replicas, bindings, assignments, admission counters, and journal	Admin job writes catalog; gateway routes; lifecycle service updates replica state
Connector host	Provider-facing runtime, checkpoints, replay, outbox, and product-specific durable state	Owned by the admitted host boundary

Use PostgreSQL for a clustered core deployment. `PostgresRuntimeStore` installs and verifies immutable schema migrations at connection time, then supplies the complete runtime store set. A file-backed `storage_dir` is a single-host boundary and is not a clustered scheduler or shared failover store.

Install connector-registry migrations with administrative credentials before starting data-plane services. The data-plane connection refuses a schema whose version differs from the required version. Reconcile least-privilege grants after migration so removed permissions do not survive from an older release.

Give each logical state owner a separately restorable database and credential scope. One PostgreSQL service may host them, but do not point a connector host at the core runtime or registry database as an undocumented shortcut.

Before rollout:

1. take named, time-stamped backups;
2. record the database and schema revision represented by each backup;
3. verify restoration in an isolated environment;
4. measure recovery point and recovery time against the deployment objective;

5. confirm that restoration preserves leases, fences, idempotency,

transactions, replay, assignments, and provider operation references.

A database connection check is not a recovery test. Use controlled in-flight and unknown-outcome cases without affecting production provider data.

Establish identities and secrets

Follow Identity and trust to assign one stable identity to every security boundary. At minimum, record:

- external client issuer, subject, audience, scopes, and tenant mapping;
- `getaip-server` service principal, trust domain, signing DID, and trusted client DIDs;
- MCP protected-resource identifier, authorization server, introspection

identity, audience, required scopes, and browser origins when MCP is public;

- connector control-plane principal and signing DID;
- each host principal, signing DID, tenant, membership, and provider account;
- release-package and seven evidence-role trust roots;
- approval authority and trusted identity directory revisions.

The pinned daemon defaults to deny-all identity and approval resolvers when no deployment-owned directory is installed. Do not replace that behavior with caller-supplied identity in action input or metadata.

Store database URLs, bearer tokens, OAuth client secrets, signing seeds, provider credentials, NATS passwords, and private keys in owner-controlled, bounded files or an equivalent mounted secret boundary. Keep only opaque secret provider and credential revision references in registry and configuration records.

Separate these signing roles:

- external client request identity;
- gateway response, callback, and fleet dispatch identity;
- lifecycle control-plane response identity;
- connector-host request and event identity;
- admission-package release authority;
- each independent supply-chain evidence authority.

Do not reuse one seed merely because the components share a trust domain.

Configure the published profiles

Expose only the profiles that the deployment claim includes. The profiles, transports, and connectors page defines their protocol roles.

For native HTTP:

- require signed envelopes, or configure a server-owned bearer identity at the

trusted edge;

- bind tenant and scopes after authentication;
- publish an HTTPS origin;
- bound request bodies, responses, streams, queues, and callbacks.

For MCP over HTTP:

- publish protected-resource metadata;
- advertise the accepted authorization server and resource identifier;
- verify token issuer, audience, activity, and required scopes through the

configured verifier;

- allow only exact browser origins when browser access is required;
- keep stdio as a local process boundary rather than a public network edge.

For A2A:

- publish the agent card from the external HTTPS origin;
- protect task operations through the same trusted AIP identity boundary;
- encrypt stored push credentials and restrict callback destinations.

For NATS, when selected:

- configure the server URL, trust domain, service, version, and queue group;
- supply username and owner-controlled password file together;
- enforce account, subject, and TLS policy outside `getaip-server`;
- include the NATS listener in the readiness and failure plan.

Do not expose every profile merely because the binary implements it. Fewer published surfaces reduce authentication, compatibility, and incident scope.

Install registry and lifecycle services

Use this order:

1. Install and verify the registry schema under the short-lived admin role.
2. Reconcile gateway data-plane and lifecycle grants.
3. Verify that data-plane credentials can read the schema version and only the

records their role requires.

4. Start the connector lifecycle service behind its TLS boundary.
5. Record its signer DID and distribute that public identity to admitted hosts.
6. Verify lifecycle `/health`, `/ready`, and bounded `/metrics` separately.
7. Plan each signed connector admission package without writing.
8. Apply only the exact reviewed package revision and digest.

The lifecycle service exposes one signed control route for register, heartbeat, drain, and offline commands. It deliberately retains no catalog-administration pool. A host must already have a pre-provisioned replica identity; registration does not create an arbitrary connector or tenant binding.

Keep tenant capability bindings disabled until the matching host artifact is running, ready, and verified.

Start the core before product traffic

Start one `getaip-server` replica with:

- the exact gateway artifact and HTTPS public origin;

- shared PostgreSQL runtime credentials;
- trusted signer, identity, approval, and profile policy revisions;
- connector-registry data-plane credentials;
- a fleet signing seed and host trust policy;
- bounded callback, reconciliation, remote-admission, event-ingress, and

transport limits;

- optional authenticated NATS configuration;
- no bundled product connector code.

At startup, fleet configuration is all-or-none. The source requires a registry URL, an owner-only signing seed, and either explicit allowed hosts or trust in admitted registry endpoints. Partial fleet configuration fails rather than starting an ambiguous gateway.

Do not add a second core replica until the first has completed runtime schema setup, established its worker, and passed the deployment checks. Then add replicas against the same logical runtime database, identity and policy revision, registry, public origin, and NATS queue group where applicable.

Start admitted connector hosts

For each required connector boundary:

1. Confirm the registry contains the intended active type and version,

tenant-owned instance, offline replica, disabled binding, artifact digest, manifest digest, endpoint, DID, topology, and capacity.

2. Mount the host database URL, signing seed, gateway and control-plane trust,

product configuration, provider credential files, and revision policy.

3. Start one replica from the exact admitted artifact.
4. Require host `/ready`, a current registry lease, and the expected manifest.
5. Exercise deterministic and non-mutating provider checks.
6. Enable one reviewed tenant capability binding at a new policy revision.
7. Route one approved low-risk request through the public client path.
8. Expand tenants, capabilities, replicas, or capacity only after the first

binding meets its observation window.

The common host exposes process health, readiness, metrics, manifest, and signed native AIP execution. Host readiness includes a valid lease, durable storage, connector health, and a non-draining state. It does not prove that the provider will accept every operation or that the connector is qualified.

When replacing a bundled connector, follow Migrate bundled connectors to the fleet so the old and new paths do not execute the same mutation.

Define readiness and traffic gates

Use separate gates rather than one aggregate health check:

Gate	What it establishes	What it does not establish
Process health	Listener and process identity respond	Storage, workers, fleet capacity, or provider access

Gate	What it establishes	What it does not establish
Core readiness	Storage probe, runtime worker, required NATS listener, and required local modules are ready	A ready replica for every fleet capability
Fleet summary	Registry maintenance state, leases, capacity, assignments, and pool signals are observable	Provider correctness or tenant authorization
Host readiness	Lease, storage, connector probe, and drain state permit work	Live capability qualification
Tenant discovery	Enabled binding exposes an admitted capability to one tenant	Successful provider execution
Controlled action	Exact route and provider behavior succeed for one case	General availability or every capability

The current `getaip-server` `/ready` response includes a fleet snapshot when fleet services are installed, but fleet summary contents do not change the core readiness Boolean. Gate connector traffic on tenant-visible bindings and ready replica capacity in addition to core readiness.

Use `/metrics` for bounded operational signals and `/ready` for traffic admission. Do not make `/health` a load-balancer readiness check.

Roll out and verify the release

Use a bounded canary sequence:

1. Keep the previous artifact set and database backups available.
2. Start one core canary with no new connector binding.
3. Verify authentication rejection and acceptance paths for each published profile.
4. Read existing action, session, approval, transaction, receipt, and event state without repeating provider work.
5. Admit and start one connector host, then enable one low-risk tenant binding.
6. Run an approved read and one controlled mutation with a unique idempotency key where the contract requires it.
7. Verify the route assignment, provider operation, stored result, event or callback, receipt, and audit correlation.
8. Exercise cancellation, timeout, retry, reconciliation, approval, and compensation only where the manifest claims support.
9. Restart one canary component and verify durable recovery without duplicate provider effects.
10. Expand replicas and traffic in measured stages.

Monitor at least:

- core and host readiness transitions;
- runtime worker and optional NATS listener state;
- registry maintenance failures, ready replicas, active assignments, and pool pressure;

- remote admission active, queued, byte, tenant, and rejection signals;
- connector lease, heartbeat, drain, and offline transitions;
- action queue age, retries, dead letters, callbacks, transactions, and

reconciliation;

- provider rate limits, latency, errors, unknown outcomes, and webhook replay;
- authentication, authorization, signature, and credential-revision failures.

Define alert thresholds from measured deployment objectives. Source defaults are safety bounds, not production capacity recommendations.

Stop on unsafe evidence

Observation	Immediate decision
Artifact, manifest, schema, or package digest differs	Stop rollout and identify the unreviewed input
Database schema or migration checksum differs	Keep traffic off and restore the reviewed migration path
Core ready but no eligible connector route exists	Keep that binding disabled and inspect registry and host state
Host health passes but readiness fails	Inspect lease, storage, trust, connector health, and drain state
A request reaches the wrong tenant or provider account	Disable the binding and rotate or correct credentials
Provider mutation outcome is unknown	Reconcile by provider operation ID; do not retry through another path
Duplicate provider effect appears	Pause mutations and preserve idempotency, assignment, and audit evidence
Identity or authority directory is stale, expired, or revoked	Keep the affected edge closed and install a reviewed revision
Backup restoration cannot meet the objective	Do not expand production traffic

Do not repair uncertainty by deleting assignments, idempotency records, replay claims, transaction state, or provider operation IDs. These are the controls needed to decide whether another side effect is permitted.

Roll back by ownership boundary

Rollback future traffic in the reverse order of admission:

1. Pause new mutations for affected tenants and capabilities.
2. Disable new or changed tenant bindings.
3. Drain connector hosts and reconcile every active or unknown provider

operation under the replica that created it.

4. Restore provider ingress only after pending host events and callbacks are

drained or retained.

5. Mark connector replicas offline and stop the new host artifacts.
6. Remove the canary core replica from traffic.
7. Restore the previous core artifact and configuration against a compatible

runtime schema.

8. Restore a database only when the rollback plan accounts for provider effects

and records created after the backup.

9. Verify identity, state reads, routing, and one non-mutating operation before

resuming mutations.

A binary rollback does not reverse a booking, message, update, provider job, or other external effect. Use the capability's governed compensation or business recovery procedure when one exists.

Retain the failed release unit, logs, traces, assignments, provider identifiers, database revisions, and operator decisions for review. Revoke an artifact, signing identity, or provider credential when the incident affects its integrity; traffic removal alone does not invalidate it.

Record the accepted production boundary

Accept the deployment only when the retained record identifies:

- exact source, artifact, manifest, schema, configuration, trust, and provider

revisions;

- topology, tenant, capability, connector, and account scope;
- admission, conformance, security, recovery, restart, load, and live-provider

evidence that actually applies;

- known exclusions and expired or missing evidence;
- observed canary and rollout times;
- recovery and rollback results;
- the owner and expiry date of the production claim.

Describe each evidence level accurately. Implementation source, conformance, artifact qualification, deployment observation, and live-provider verification answer different questions and cannot substitute for one another.

Related documentation

- Examples and reference deployments
- Actions and sessions
- Approvals and policy
- Connector catalog