Run the CrewAI connector quickstart

In this tutorial, you will start the repository's deterministic CrewAI product path and route one governed run through the standalone Rust host and Python sidecar.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/crewai/getting-started/quickstart.md

The default product sidecar uses the frozen CrewAI `1.15.5` package set. Its network-free qualification model emits deterministic real CrewAI events.

This is a controlled learning exercise. It is not a production deployment, external-model test, or complete qualification run.

What you will verify

By the end, you will have observed:

- source-derived Rust-host and Python-sidecar image identities;
- registry admission before the CrewAI host starts;
- agreement between one host descriptor and one sidecar registry entry;
- a signed run held for tenant-policy approval;
- the same Action resumed with its stable idempotency key;
- one terminal real-CrewAI result;
- durable status and event replay keyed by the original Action ID;
- cleanup limited to the named tutorial project.

You will not exercise batch, replay, train, test, knowledge, memory reset, cancellation races, storage outage, gateway failover, or an external provider.

Prerequisites

You need:

- a clean `aip-core` checkout at the reviewed source revision;
- Docker Engine, Docker Compose v2, and BuildKit;
- network access required by the source-owned image build;
- enough CPU, memory, and disk for all product-profile images;
- TCP port `18443` free on loopback;
- a POSIX-compatible shell, `lsof`, `grep`, and `sed`;
- authority to remove the named project after reviewing its state.

Run every command from the `aip-core` repository root. The products profile builds all product hosts even though this tutorial starts the CrewAI path.

The Compose project is `aip-connector-fleet-qualification`. Stop if another investigation or retained run owns that project or its volumes.

1. Verify source and port ownership

Set and verify the reviewed AIP revision:

```
SH
export AIP_SOURCE_REVISION=97be86e9efedf07ecf1783b03800f683f107fb04

test "$(git rev-parse HEAD)" = "$AIP_SOURCE_REVISION"
test -z "$(git status --porcelain)"
```

Use a separate clean checkout when either check fails. Never hide or discard local work to satisfy this tutorial.

Confirm the required tools and fixed edge port:

```
SH
docker info >/dev/null
docker compose version

command -v lsof >/dev/null
if lsof -nP -iTCP:18443 -sTCP:LISTEN | grep -q .; then
  echo "Port 18443 is already in use" >&2
  exit 1
fi
```

Expected result: the port check prints nothing. Identify an existing listener before changing it.

2. Prepare controlled artifacts

Run the source-owned preparation script:

```
SH
./deploy/connector-fleet/prepare.sh
test -s deploy/connector-fleet/.env.qualification
```

Expected result: preparation ends with `connector fleet preparation: PASS`. The owner-only environment file records source, image, release, and revision identities selected by Compose.

The sidecar build uses the repository's default `Dockerfile` and frozen `uv.lock`. This product profile does not select the separate source-overlay `Dockerfile.upstream` path.

Define one helper for the source-owned profile:

```
SH
compose() {
  docker compose \
    --profile products \
    --env-file deploy/connector-fleet/.env.qualification \
    -f deploy/connector-fleet/compose.yml \
    "$@"
}

compose ps --all
```

Stop if the fixed project contains state that must be retained.

3. Start the CrewAI path

Start the host and its declared dependencies:

```
SH
compose up --detach --wait --wait-timeout 300 crewai-acme-a
compose ps --all
```

Expected result: the CrewAI sidecar, host, both gateways, control plane, edge, and PostgreSQL are healthy. Initialization, migration, grant, and admission jobs may be exited with status `0`.

The topology mounts one durable sidecar volume at `/var/lib/aip-crewai`. It does not start two active sidecars over that journal.

Do not bypass a failed `admit-crewai` job or start the Rust host under a different identity.

4. Verify descriptor and registry agreement

The generated host descriptor admits `support-qualification` with all ten operations. The sidecar registry constructs the same crew ID.

Inspect only public generated configuration:

```
SH
compose exec -T crewai-acme-a \
  sh -c 'test -r /fleet-state/public/crewai-crews.json'

compose exec -T crewai-sidecar \
  python -c 'import importlib.metadata; print(importlib.metadata.version("crewai"))'
```

Expected result: the descriptor file is readable and the sidecar reports `1.15.5`.

Do not print the sidecar token or signing material. Host readiness already checks authenticated sidecar health, durable journal presence, crew IDs, and operation coverage.

5. Request one crew run

Create a private directory for tutorial evidence:

```
SH
EVIDENCE_DIR=$(mktemp -d "${TMPDIR:-/tmp}/aip-crewai-quickstart.XXXXXX")
chmod 700 "$EVIDENCE_DIR"
printf 'Tutorial evidence: %s\n' "$EVIDENCE_DIR"
```

Keep one Action ID, key, and input for the entire run:

```
SH
ACTION_ID=act_crewai_quickstart_run_001
IDEMPOTENCY_KEY=crewai-quickstart-run-v1
INPUT='{"case_id":"AIP-CREWAI-QUICKSTART"}'
PENDING="$EVIDENCE_DIR/run-pending.json"
```

Submit the run with the source-owned caller identity:

```
SH
compose run --rm --no-deps \
  --entrypoint /usr/local/bin/aipctl \
  -e AIPCTL_NATIVE_PRINCIPAL_ID=service:aipctl:qualification-acme \
  -e AIPCTL_NATIVE_TRUST_DOMAIN=fleet.test \
  -e AIPCTL_NATIVE_SIGNING_SEED_FILE=/fleet-state/secrets/client-acme-signing-seed.hex \
  -e AIPCTL_NATIVE_PEER_DID_FILE=/fleet-state/public/gateway.did \
  -e AIPCTL_NATIVE_TLS_CA_FILE=/caddy-data/caddy/pki/authorities/local/root.crt \
  qualification action call \
  https://aipd.fleet.test:8443 \
  cap:crewai:support-qualification \
  --action-id "$ACTION_ID" \
  --idempotency-key "$IDEMPOTENCY_KEY" \
  --input "$INPUT" >"$PENDING"

grep -q '"status": "pending_approval"' "$PENDING"
```

Expected result: the Action is pending approval. The sidecar has not been authorized to start the crew.

Do not change any of the three stable values after this step.

6. Review and approve the run

Extract and validate the approval identity and policy hash:

```
SH
approval_id=$(sed -n \
  's/^[[:space:]]*"approval_id": "\([^"]*\)",\{0,1\}$/\1/p' \
  "$PENDING" | head -n 1)
policy_hash=$(sed -n \
  's/^[[:space:]]*"policy_hash": "\([^"]*\)",\{0,1\}$/\1/p' \
  "$PENDING" | head -n 1)

case "$approval_id" in
  appr_[A-Za-z0-9_]*) ;;
  *) echo "Unexpected approval ID" >&2; exit 1 ;;
esac

test "${#policy_hash}" -eq 64
printf '%s' "$policy_hash" | grep -Eq '^[0-9a-f]{64}$'
```

Review the pending record. Stop if its Action, capability, input snapshot, or policy differs from the request above.

Create one signed approval decision:

```
SH
decided_at=$(date -u +%Y-%m-%dT%H:%M:%SZ)
decision=$(printf \
  '{"approval_id":"%s","decision":"approved","approver":{"id":"human:qualification-
approver","kind":"human"},"decided_at":"%s","reason":"Approved for the CrewAI quickstart fixture","con
straints":[],"evidence":[],"decision_id":"decision:%s:approved","policy_hash":"%s","authority_path":[]
}' \
  "$approval_id" "$decided_at" "$approval_id" "$policy_hash")

compose run --rm --no-deps \
  --entrypoint /usr/local/bin/aipctl \
  -e AIPCTL_NATIVE_PRINCIPAL_ID=human:qualification-approver \
  -e AIPCTL_NATIVE_TRUST_DOMAIN=fleet.test \
  -e AIPCTL_NATIVE_SIGNING_SEED_FILE=/fleet-state/secrets/approval-acme-signing-seed.hex \
  -e AIPCTL_NATIVE_PEER_DID_FILE=/fleet-state/public/gateway.did \
  -e AIPCTL_NATIVE_TLS_CA_FILE=/caddy-data/caddy/pki/authorities/local/root.crt \
  qualification approval decide \
  https://aipd.fleet.test:8443 \
  "$decision" >"$EVIDENCE_DIR/approval-decision.json"

grep -q '"kind": "aip.approval.granted"' \
  "$EVIDENCE_DIR/approval-decision.json"
grep -q '"kind": "aip.action.resumed_result"' \
  "$EVIDENCE_DIR/approval-decision.json"
```

Approval authorizes this exact controlled run. It does not make model or tool effects retry-safe.

7. Read the completed Action

Repeat the exact Action to retrieve its durable result:

```
SH
COMPLETED="$EVIDENCE_DIR/run-completed.json"

compose run --rm --no-deps \
  --entrypoint /usr/local/bin/aipctl \
  -e AIPCTL_NATIVE_PRINCIPAL_ID=service:aipctl:qualification-acme \
  -e AIPCTL_NATIVE_TRUST_DOMAIN=fleet.test \
  -e AIPCTL_NATIVE_SIGNING_SEED_FILE=/fleet-state/secrets/client-acme-signing-seed.hex \
  -e AIPCTL_NATIVE_PEER_DID_FILE=/fleet-state/public/gateway.did \
  -e AIPCTL_NATIVE_TLS_CA_FILE=/caddy-data/caddy/pki/authorities/local/root.crt \
  qualification action call \
  https://aipd.fleet.test:8443 \
  cap:crewai:support-qualification \
  --action-id "$ACTION_ID" \
  --idempotency-key "$IDEMPOTENCY_KEY" \
  --input "$INPUT" >"$COMPLETED"

grep -q '"status": "completed"' "$COMPLETED"
grep -q 'case triaged' "$COMPLETED"
```

Expected result: the signed result is terminal and contains the deterministic CrewAI output. The repeated request observes the same AIP intent.

If the result remains unknown, preserve the original identities. Do not create a replacement run.

8. Read durable sidecar status

Use a new read Action that targets the original run Action ID:

```
SH
compose run --rm --no-deps \
  --entrypoint /usr/local/bin/aipctl \
  -e AIPCTL_NATIVE_PRINCIPAL_ID=service:aipctl:qualification-acme \
  -e AIPCTL_NATIVE_TRUST_DOMAIN=fleet.test \
  -e AIPCTL_NATIVE_SIGNING_SEED_FILE=/fleet-state/secrets/client-acme-signing-seed.hex \
  -e AIPCTL_NATIVE_PEER_DID_FILE=/fleet-state/public/gateway.did \
  -e AIPCTL_NATIVE_TLS_CA_FILE=/caddy-data/caddy/pki/authorities/local/root.crt \
  qualification action call \
  https://aipd.fleet.test:8443 \
  cap:crewai:support-qualification:status \
  --action-id act_crewai_quickstart_status_001 \
  --input '{"run_action_id":"act_crewai_quickstart_run_001"}' \
  >"$EVIDENCE_DIR/run-status.json"

grep -q '"status": "completed"' "$EVIDENCE_DIR/run-status.json"
grep -q 'act_crewai_quickstart_run_001' "$EVIDENCE_DIR/run-status.json"
```

This reads the sidecar journal. It does not execute the crew again.

9. Replay the event journal

Read events from cursor zero:

```
SH
compose run --rm --no-deps \
  --entrypoint /usr/local/bin/aipctl \
  -e AIPCTL_NATIVE_PRINCIPAL_ID=service:aipctl:qualification-acme \
  -e AIPCTL_NATIVE_TRUST_DOMAIN=fleet.test \
  -e AIPCTL_NATIVE_SIGNING_SEED_FILE=/fleet-state/secrets/client-acme-signing-seed.hex \
  -e AIPCTL_NATIVE_PEER_DID_FILE=/fleet-state/public/gateway.did \
  -e AIPCTL_NATIVE_TLS_CA_FILE=/caddy-data/caddy/pki/authorities/local/root.crt \
  qualification action call \
  https://aipd.fleet.test:8443 \
  cap:crewai:support-qualification:events \
  --action-id act_crewai_quickstart_events_001 \
  --input '{"run_action_id":"act_crewai_quickstart_run_001","cursor":0}' \
  >"$EVIDENCE_DIR/run-events.json"

grep -q '"event": "completed"' "$EVIDENCE_DIR/run-events.json"
```

Expected result: the bounded response includes ordered stored events and a next cursor. This is journal replay, not live streaming.

Diagnose before cleanup

Capture bounded state first:

```
SH
compose ps --all
compose logs --no-color --tail 200 \
  aipd aipd-b control-plane crewai-acme-a crewai-sidecar
```

Symptom	First check
Preparation cannot reach Docker	Repair the daemon, then repeat source checks
Port `18443` is occupied	Identify its owner without stopping it
Sidecar is unhealthy	Token file, registry, journal, lock, and pinned image
Host is unhealthy	Sidecar health, descriptor crew IDs, and operation policy

Symptom	First check
Admission exits nonzero	Preserve the job log; do not bypass admission
Run never leaves approval	Decision identity, policy hash, and Action correlation
Run fails	Sidecar log by Action ID and terminal journal event
Status returns not found	Exact original Action ID and sidecar state volume
Event cursor is rejected	Use a cursor within the retained event journal

The tutorial does not authorize printing credentials, weakening TLS, deleting the journal, editing approval state, or bypassing the gateway.

Stop and remove the tutorial project

Only after confirming that no evidence or state must be retained, remove the named project's containers, network, and volumes:

```
SH
compose down --volumes --remove-orphans --timeout 30
test -z "$(compose ps --all --quiet)"
```

Volume removal permanently deletes PostgreSQL, generated identities, the sidecar journal, and training artifacts in this project. It does not prune unrelated Docker resources.

Related documentation

- CrewAI connector overview
- AIP quickstart
- Connector fleet quickstart
- Deploy the connector fleet
- Errors and retry decisions