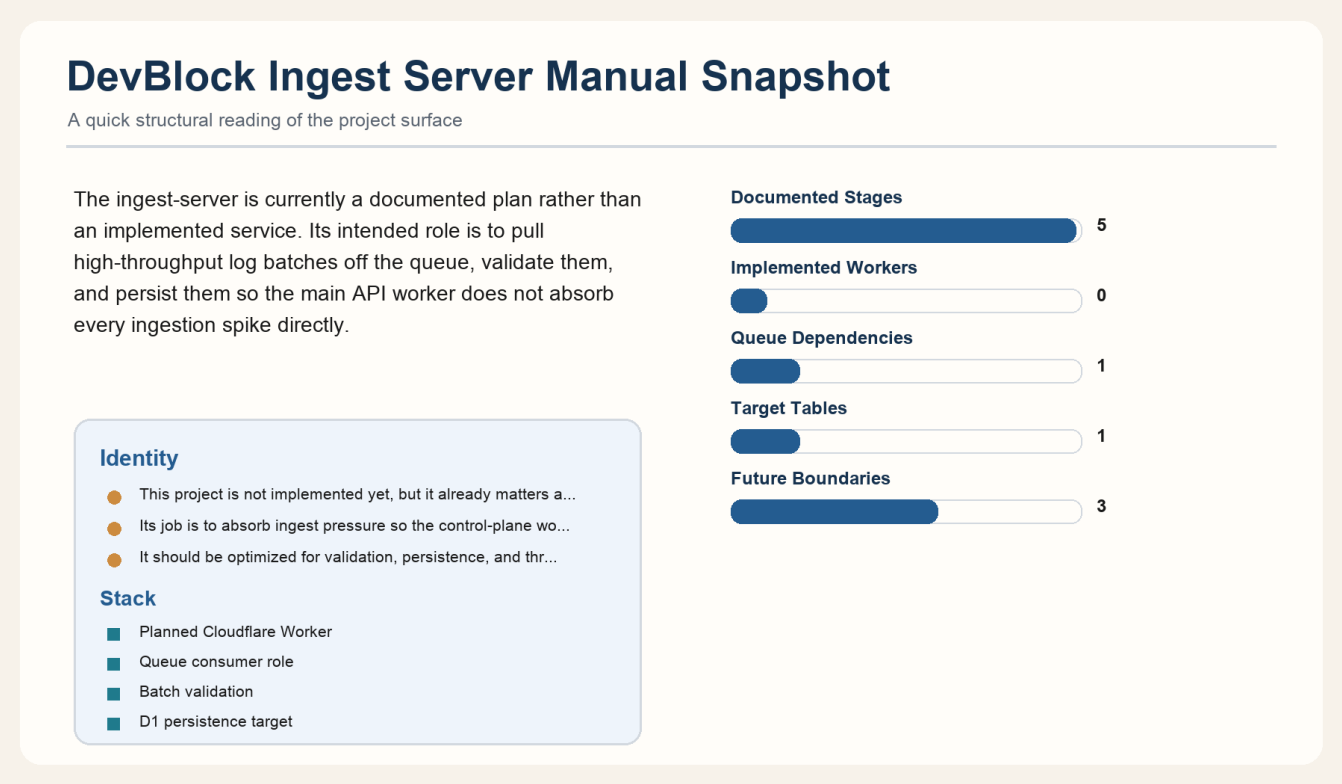
DevBlock Ingest Server Manual

Planned queue-consumer split for high-throughput telemetry ingestion

The ingest-server is currently a documented plan rather than an implemented service. Its intended role is to pull high-throughput log batches off the queue, validate them, and persist them so the main API worker does not absorb every ingestion spike directly.



DevBlock Ingest Server Manual visual snapshot

What this project is for

The ingest-server is currently a documented plan rather than an implemented service. Its intended role is to pull high-throughput log batches off the queue, validate them, and persist them so the main API worker does not absorb every ingestion spike directly.

> **Primary audience: Use this when planning the queue-consumer split or deciding what should leave the main API worker over time.**

How to identify it quickly

- This project is not implemented yet, but it already matters as an architectural boundary.
- Its job is to absorb ingest pressure so the control-plane worker does not carry every burst directly.
- It should be optimized for validation, persistence, and throughput, not broad product logic.
- A good design here reduces coupling in the main worker rather than recreating it elsewhere.

Technology stack

- Planned Cloudflare Worker
- Queue consumer role
- Batch validation
- D1 persistence target

Structural counts

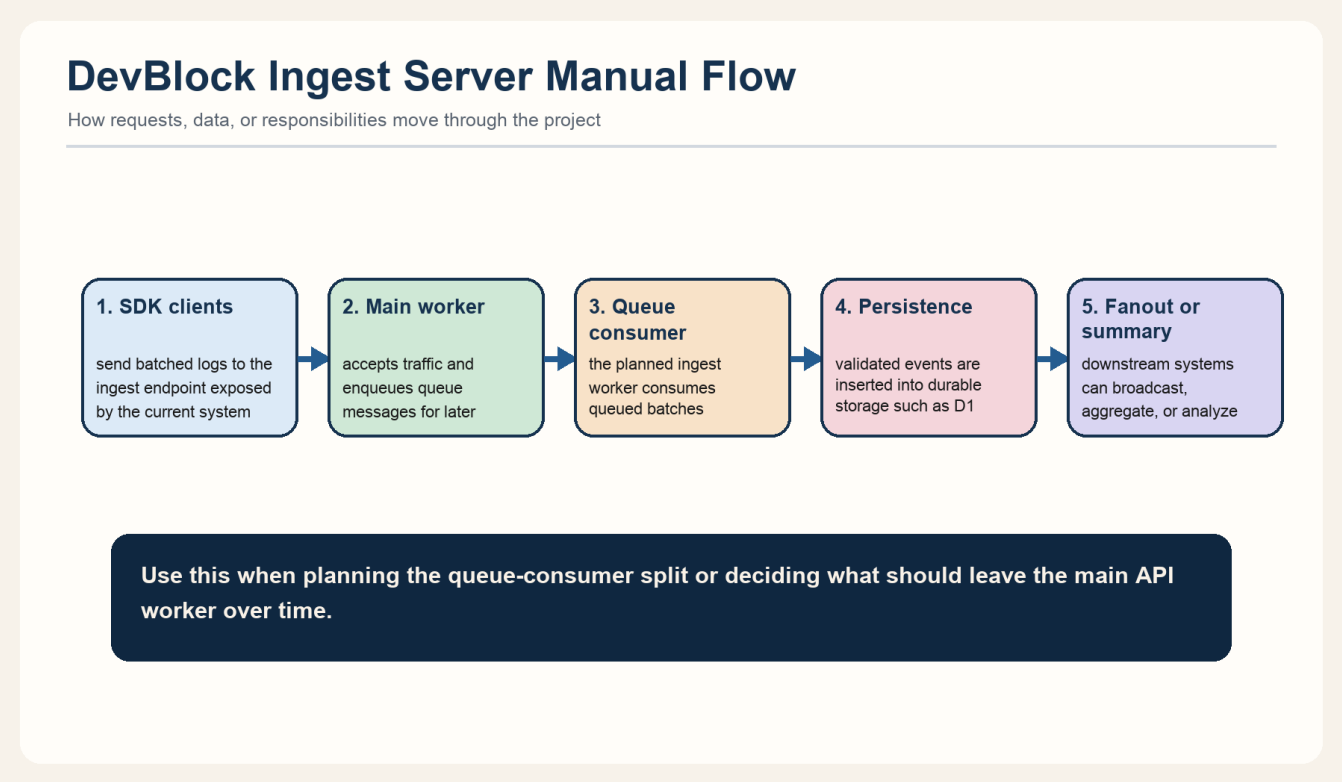
Metric	Count
Documented Stages	5
Implemented Workers	0
Queue Dependencies	1
Target Tables	1
Future Boundaries	3

Architecture reading

- This service exists to preserve backend focus by separating ingestion throughput concerns from control-plane concerns.
- Its success depends on clean responsibility boundaries more than on raw queue mechanics alone.
- It should only own the ingest lifecycle, not opportunistically absorb unrelated backend tasks.
- Documented architecture matters now because it influences how current ingest code evolves inside server-worker.

Core flow

This sequence is the shortest useful mental model for how the project behaves at runtime or during operator use.



DevBlock Ingest Server Manual responsibility or data flow

Responsibility matrix

DevBlock Ingest Server Manual Responsibility Matrix

What this project owns versus what it coordinates

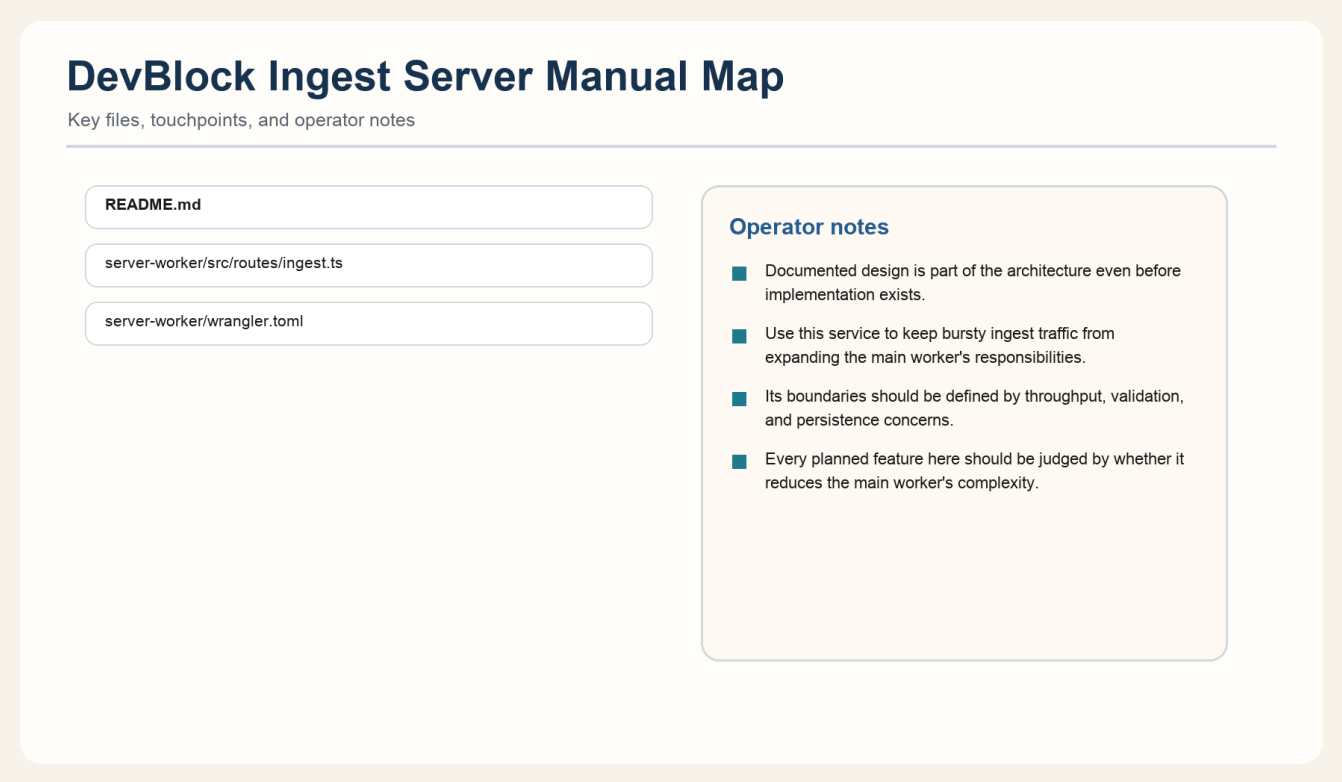
Area	Owns	Coordinates with	Failure mode
Product role	This project is not implemented yet, but it already matters as an architectural boundary.	Its job is to absorb ingest pressure so the control-plane worker does not carry every burst directly.	Because the service is planned, docs can drift from eventual implementation choices.
Main inputs	SDK clients: send batched logs to the ingest endpoint exposed by the current system	Main worker: accepts traffic and enqueues queue messages for later processing	Queue semantics, retry policy, and idempotency strategy still need explicit design.
Change surface	Read the ingest README and current worker route together before proposing a split.	Clarify queue semantics, idempotency, and failure handling before writing any implementation	The split only helps if ownership boundaries are maintained after implementation.
Operator posture	Design reviews matter more than code-level checks until the service exists.	Validate proposed boundaries against actual ingest load and current server-worker	Documented design is part of the architecture even before implementation exists.

DevBlock Ingest Server Manual ownership matrix

Key files to read first

If you are new to this project, read these files in order before making broad edits.

- README.md
- server-worker/src/routes/ingest.ts
- server-worker/wrangler.toml



DevBlock Ingest Server Manual orientation map

Change workflow

- Read the ingest README and current worker route together before proposing a split.
- Clarify queue semantics, idempotency, and failure handling before writing any implementation code.
- Use the future service to remove pressure from the main worker, not to duplicate control-plane concerns.
- Keep the queue-consumer contract narrow so validation and persistence remain the center of gravity.

Operator notes

- Documented design is part of the architecture even before implementation exists.
- Use this service to keep bursty ingest traffic from expanding the main worker's responsibilities.
- Its boundaries should be defined by throughput, validation, and persistence concerns.
- Every planned feature here should be judged by whether it reduces the main worker's complexity.

Validation posture

- Design reviews matter more than code-level checks until the service exists.
- Validate proposed boundaries against actual ingest load and current server-worker responsibilities.
- Insist on explicit retry and idempotency notes before implementation begins.

Current risks or watchpoints

- Because the service is planned, docs can drift from eventual implementation choices.
- Queue semantics, retry policy, and idempotency strategy still need explicit design.
- The split only helps if ownership boundaries are maintained after implementation.

Glossary

Term	Meaning
Ingest	The process of receiving, validating, and storing incoming telemetry or logs.
Idempotency	The property that repeated processing of the same message does not corrupt state.
Queue consumer	A worker that processes messages after they have been accepted into a queue.