AI APIs in Practice

A Visual Engineering Guide to Multimodal AI,
Agents, RAG, Evaluation, Governance, and Real-World Products

Multimodal AI | Agents | RAG | Evaluation | Governance | Real-World Products

GOVERNANCE & TRUST

Safety & Alignment | Privacy & Security | Bias & Fairness | Observability & Monitoring | Audit & Compliance

Faramarz Kowsari

Google Books Key • GGKEY: DTJJHJUDW0Y

DOI: 10.5281/zenodo.21431944

Free source code for this book is available at:
https://faramarzkowsari.github.io/ai-api-playbook/

THE AI API ECOSYSTEM

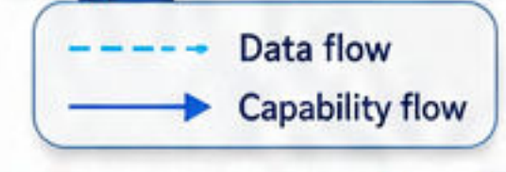
One request can activate text, tools, knowledge, sight, voice and video.

- Text APIs generate, reason and code

golden rule

- Tool APIs connect models to actions

verify



- Retrieval APIs ground answers in evidence

verify

- Media APIs understand and create multiple modalities

golden rule

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY

`import requests`
 	
`response = requests.post(`
`  "https://api.service/v1/run",`
`  json={`
`  "input": payload,`
`  "tools": tools,`
`  "data": context`
`  },`
`  headers={"Authorication": token}`
`)`
 	
`result = response.json()`
`validate(result))`
`return result`
	Customer intent + authorized context	Useful result + evidence + usage metrics	Value appears when a capability solves a recurring customer problem.	docs/book-map.md

ANATOMY OF AN AI API CALL

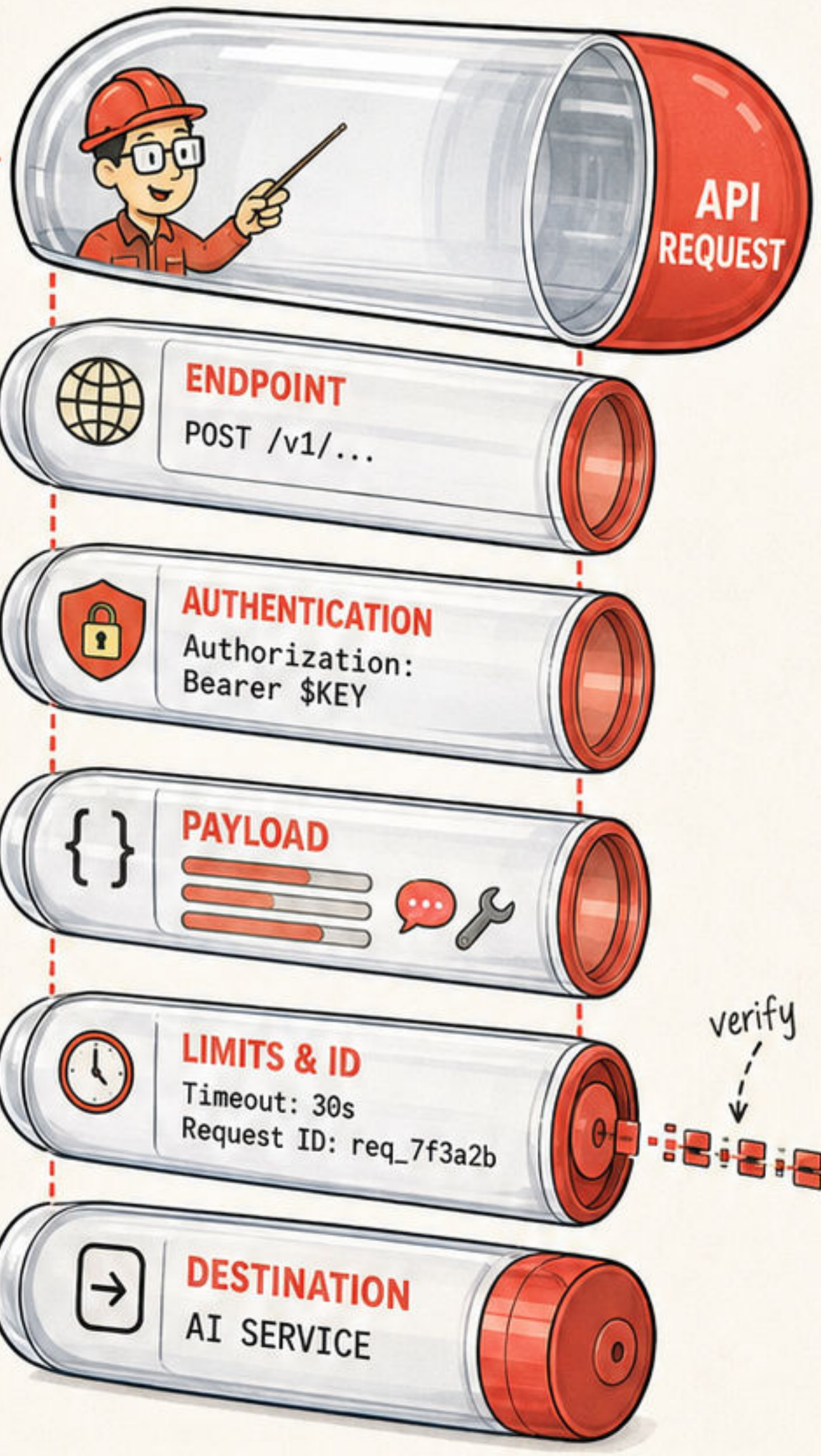
Every reliable call has identity, instructions, input, limits and a destination.

1 Endpoint chooses the service

2 Authentication proves permitted access

3 Payload carries messages, tools and constraints

4 Timeout and request ID make failures manageable



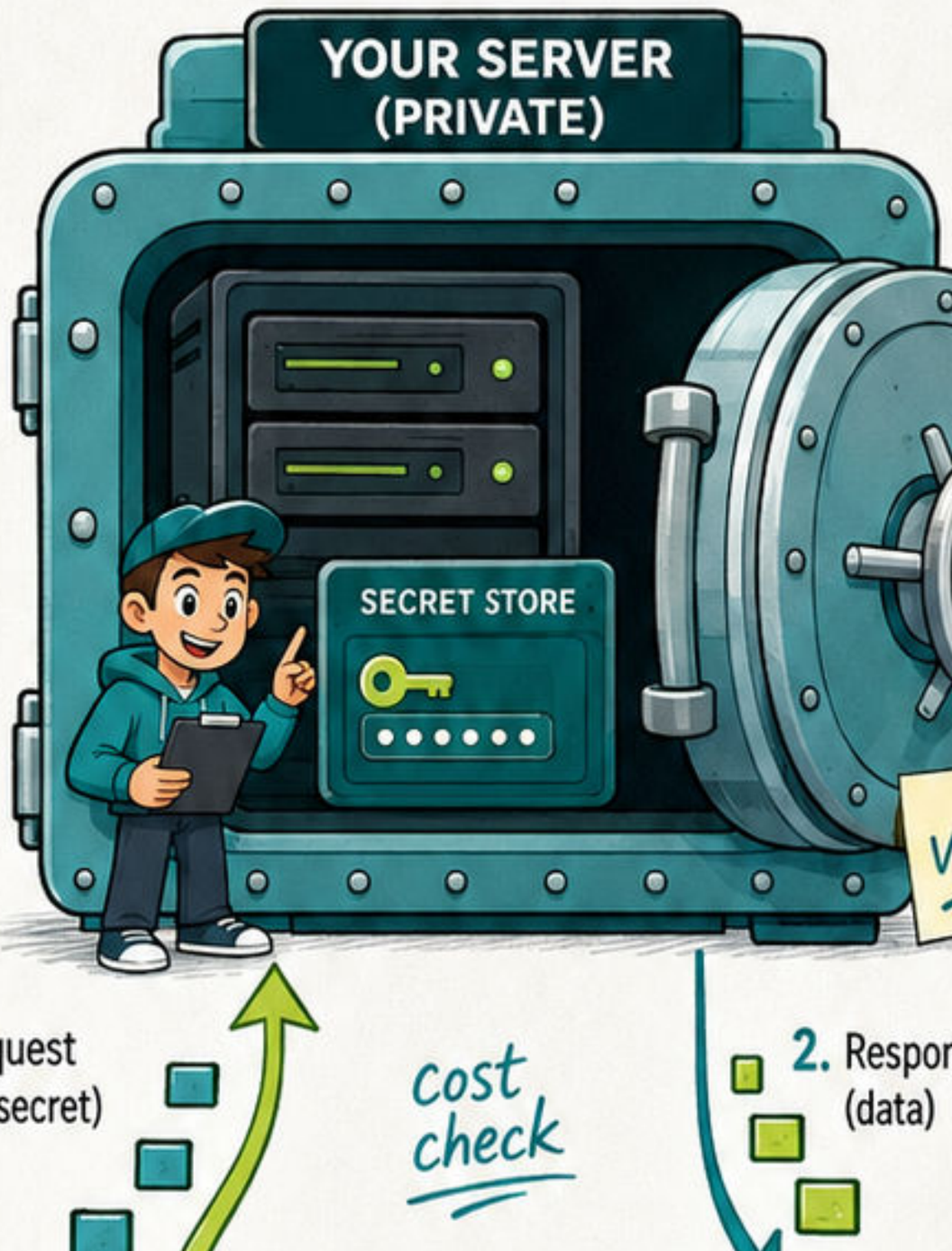
CODE	INPUT	OUTPUT	COMMERCIAL LENS

`POST /v1/...`
`Authorization: Bearer $KEY`
`Content-Type: application/json`
	Prompt, model policy, schema, metadata	Content, tool calls, usage, status, request ID	golden rule
			A reusable request wrapper reduces engineering time across products.

REPOSITORY `src/ai_api_playbook/models.py`

API KEY SECURITY

A key is not a password you type everywhere—it is a capability you must contain.



1

Store secrets in a managed secret store

2

Never commit .env or expose privileged keys to clients

3

Rotate leaked credentials immediately

4

Use least privilege, spend limits and separate environments

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY

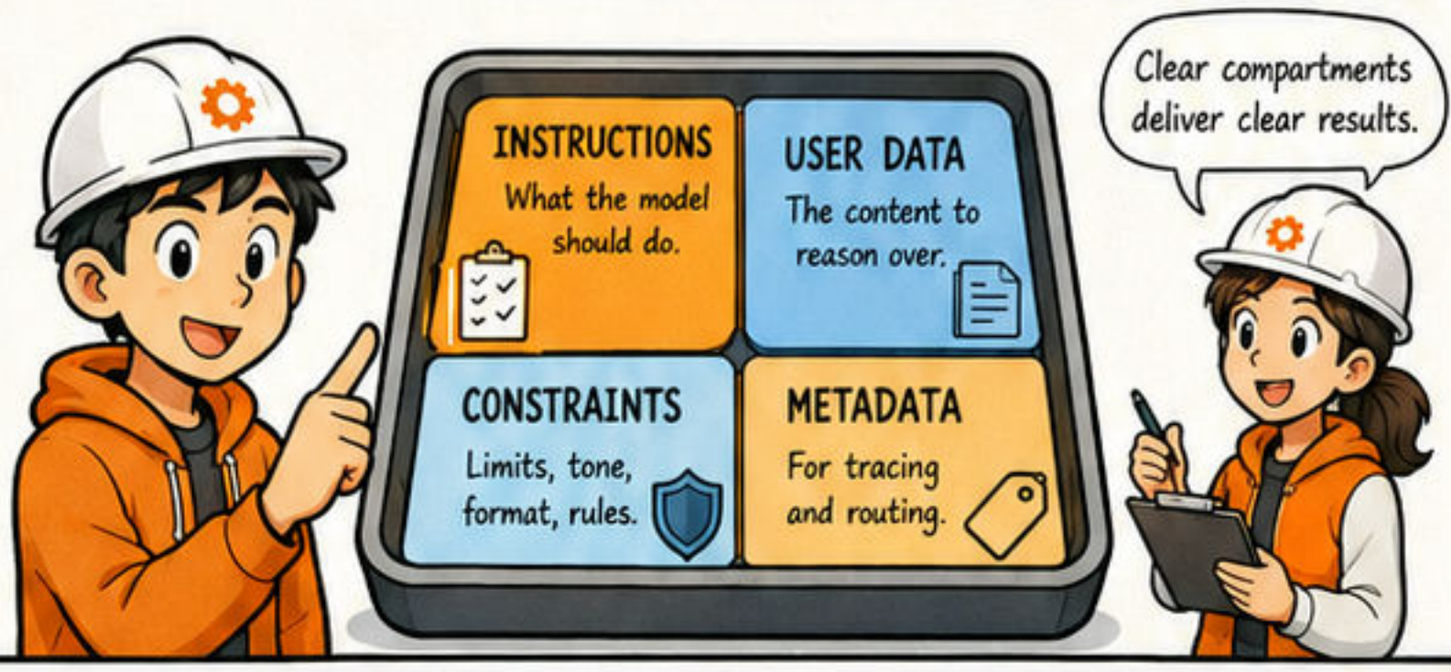
`import os`
 	
`key = os.environ[`
`"PROVIDER_API_KEY"`
`]`
	Server-side secret reference	Authenticated request without revealing the secret	Security is part of product value, not an afterthought.	SECURITY.md

THE REQUEST PAYLOAD

Good payloads remove ambiguity before the model begins.

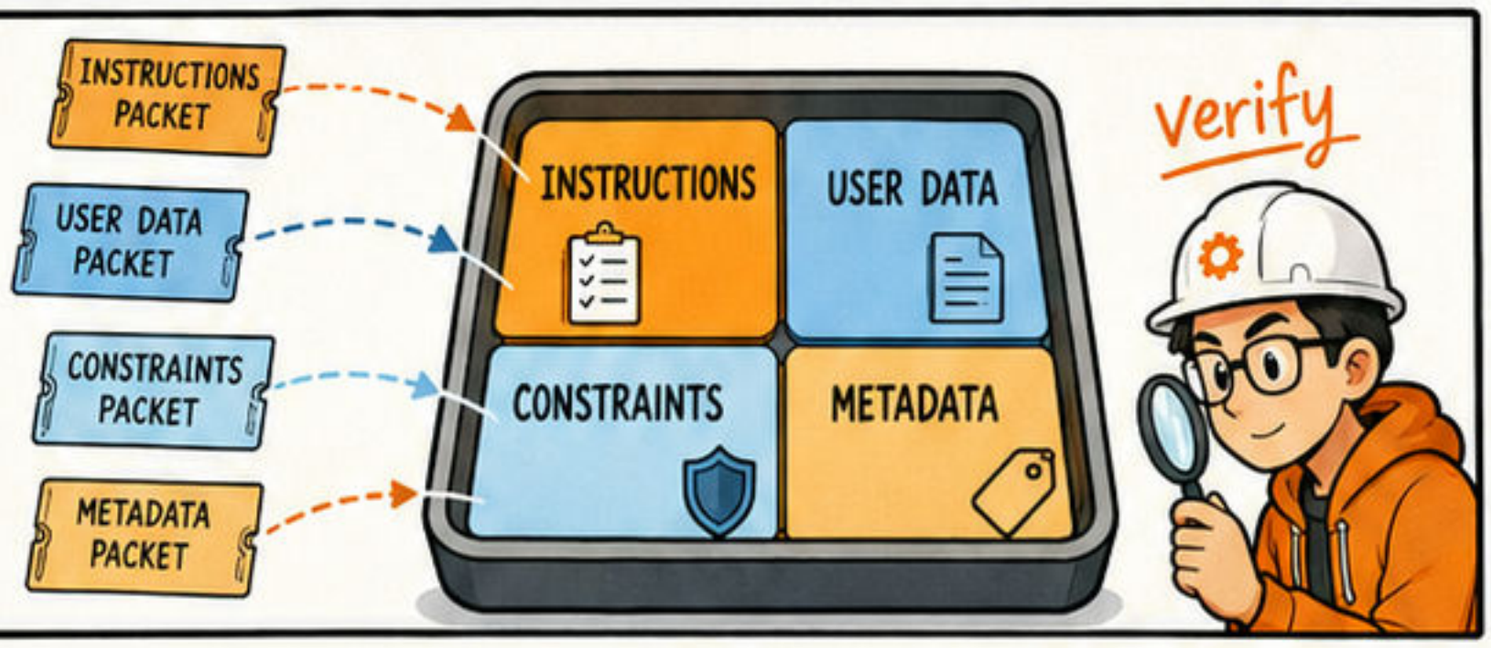
1

Think of a good payload as a bento box—each compartment has a clear job.



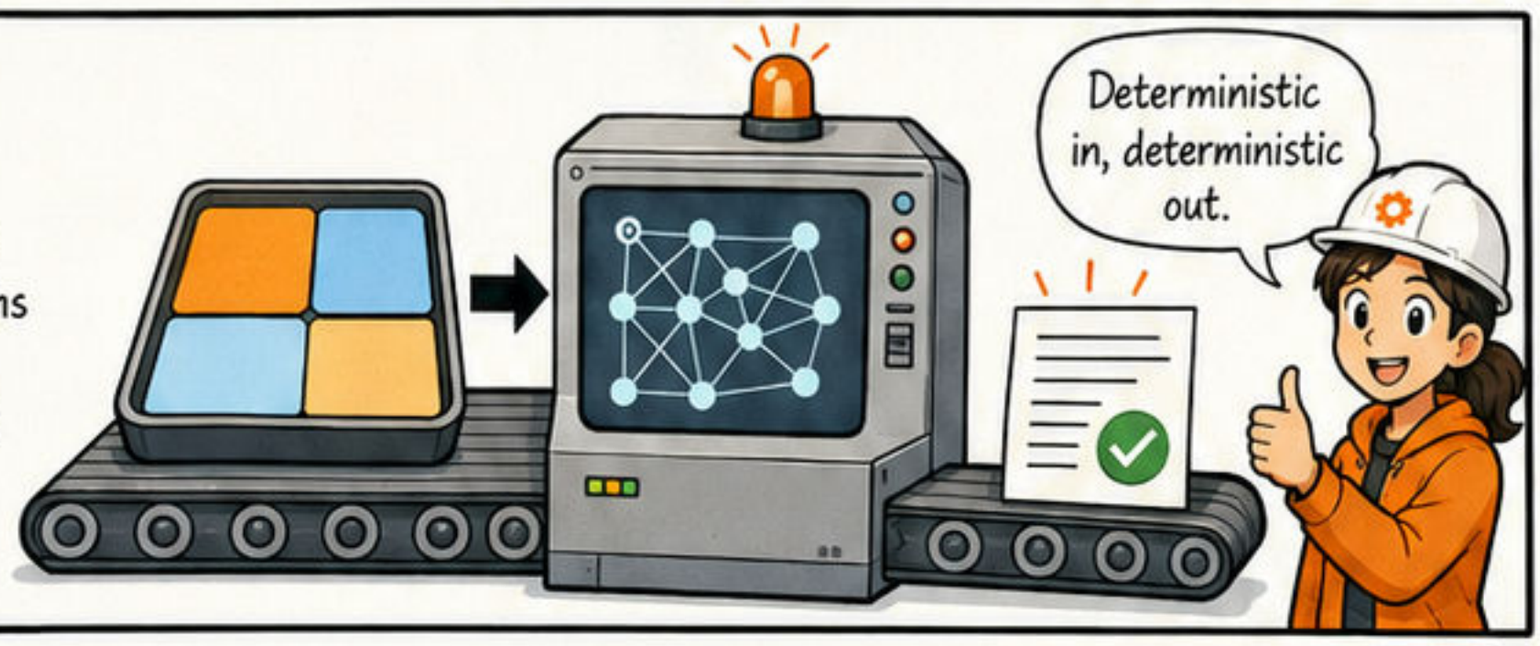
2

Pack it well. Keep roles separate and be specific.



3

The model processes the box and returns exactly what you asked for.



GOLDEN NOTE 1

Separate instructions from user data

GOLDEN NOTE 2

Set explicit output limits

GOLDEN NOTE 3

Use schemas for machine-consumed results

GOLDEN NOTE 4

Attach metadata for tracing, not sensitive content

CODE

```
{"model":"...",
 "input":"...",
 "max_output_tokens":800}
```

INPUT

OUTPUT

COMMERCIAL LENS

REPOSITORY

ENDPOINTS, METHODS & HEADERS

The URL says where; the method says what; headers say how.

golden rule

★ GET reads; POST creates or computes

★ Headers carry auth, content type and trace data

★ Path parameters identify resources

★ Query parameters tune filtering and pagination

ENDPOINT

Where the request goes.
Base URL + path + query

```
https://api.example.com/v1/responses
?model=gpt-4o&limit=10
```

DATA PACKET

METHOD

What action to perform.
Tells the server how to treat it.

POST

DATA PACKET

POST

HEADERS

How to communicate.
Metadata about the request.

```
Authorization: Bearer sk-...
Content-Type: application/json
X-Request-ID: 7f3c2d90-1a2b-4c3d
Accept: application/json
```

DATA PACKET

BODY

What data to send.
Payload for the request.

```
{
  "model": "gpt-4o",
  "prompt": "Explain APIs in simple terms.",
  "max_tokens": 500
}
```

DATA PACKET

{ }

SERVER

CODE

```
POST https://api.example.com/v1/responses
Authorization: Bearer sk-...
Content-Type: application/json
X-Request-ID: 7f3c2d90-1a2b-4c3d
Accept: application/json

{
  "model": "gpt-4o",
  "prompt": "Explain APIs in simple terms.",
  "max_tokens": 500
}
```

INPUT

Headers + JSON body

OUTPUT

HTTP status + headers + JSON body

COMMERCIAL LENS

A clean API client becomes reusable infrastructure for many paid services.

REPOSITORY

docs/architecture.md

READING THE RESPONSE

Do not confuse generated text with a completed, validated business result.

1 ★ GOLDEN RULE

Check completion status and finish reason

2 ★ GOLDEN RULE

Parse tool calls separately from text

3 ★ GOLDEN RULE

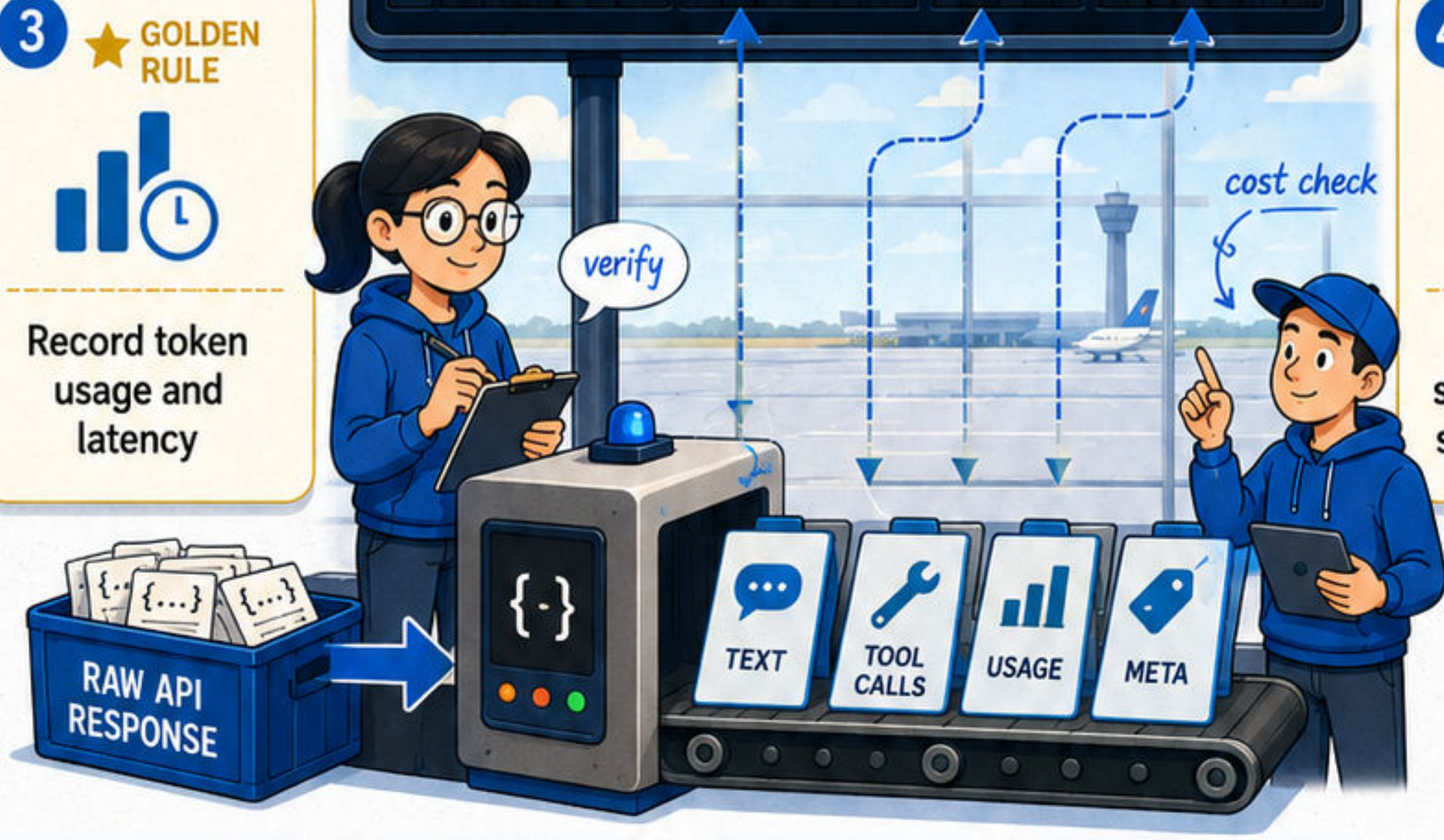
Record token usage and latency

4 ★ GOLDEN RULE

Validate structure before saving or acting

AI API ARRIVALS — INCOMING RESPONSE — 10:42

STATUS	CONTENT	USAGE	REQUEST ID
COMPLETED	TEXT	1,248 tk	req_8f3a1c7e
COMPLETED	TOOL CALLS	2,016 tk	req_3b7d9f20
IN PROGRESS	—	512 tk	req_2c1e7a91
FAILED	ERROR	256 tk	req_e9d4a5b2
COMPLETED	TEXT	1,102 tk	req_7a6b2d33



CODE

```
if response.status
    != "completed":
  raise
    RuntimeError()
```

INPUT

Provider response object

OUTPUT

Validated domain object + safe telemetry

COMMERCIAL LENS

Reliable response handling reduces support cost and failed jobs.

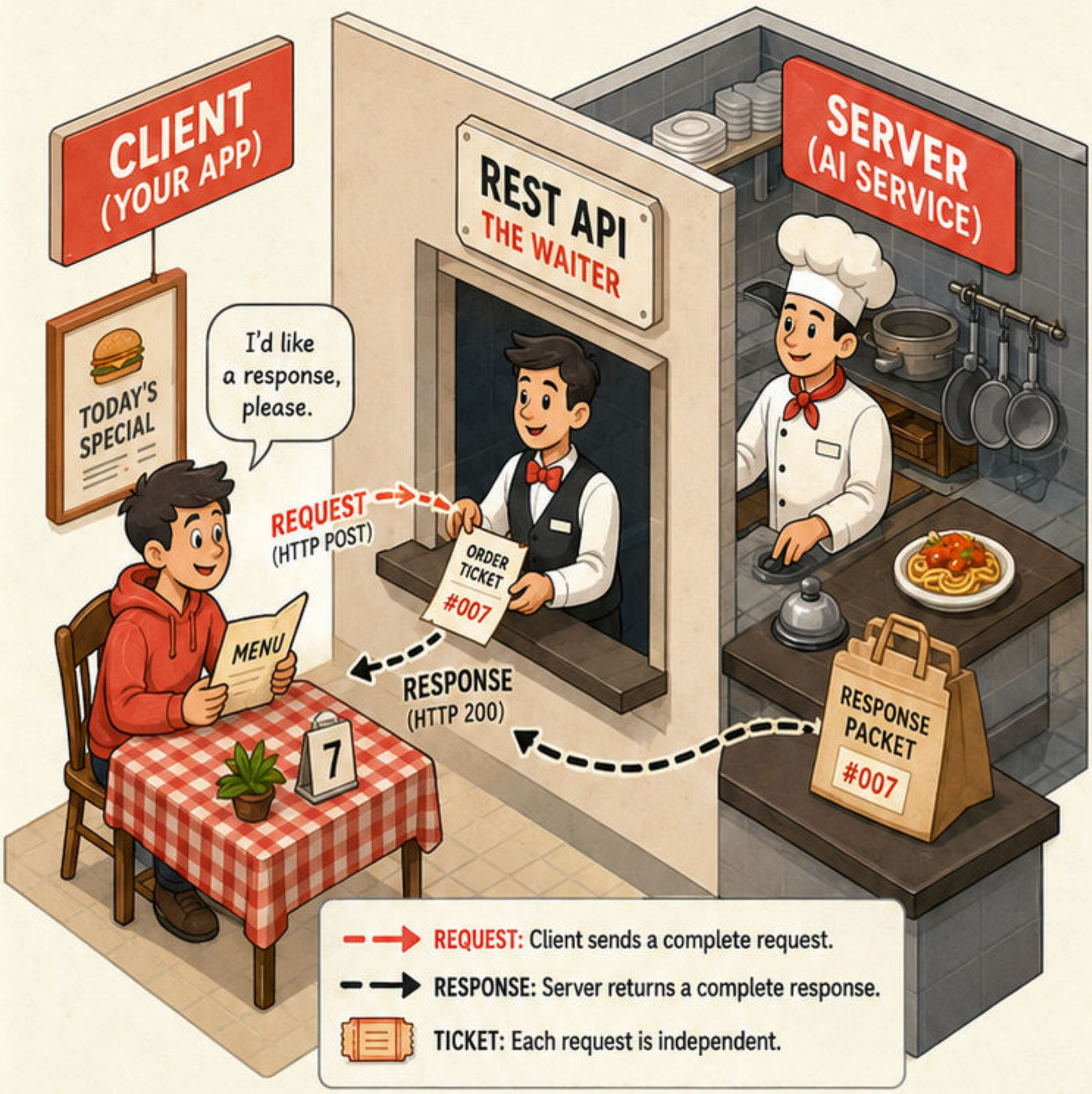
golden rule

REPOSITORY

src/ai_api_playbook/client.py

REST: THE DEFAULT TRANSPORT

Simple request/response remains ideal for short independent jobs.



golden rule

1. REST is easy to test, cache and observe
2. Use POST for generation requests
3. Set connect and read timeouts
4. Retry only safe or idempotent operations

verify

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`r = httpx.post(url, json=payload, timeout=30)`	ORDER TICKET #007 — One complete request	RESPONSE PACKET #007 — One complete response	REST is the fastest path to a sellable backend MVP.	examples/01_text/openai_responses.py

STREAMING WITH SSE

Show useful output before the model finishes the full answer.

Streaming reduces perceived latency

Process typed events, not raw text fragments

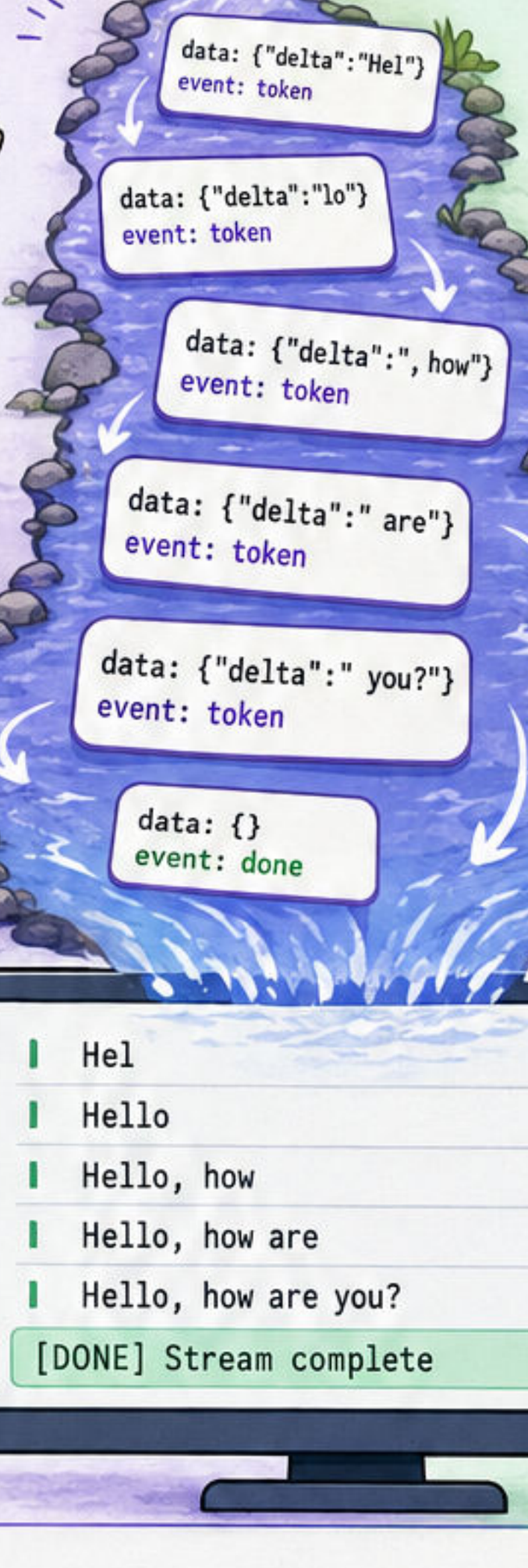
Accumulate a final canonical result

Handle disconnects and partial output explicitly

UPSTREAM
From model to stream

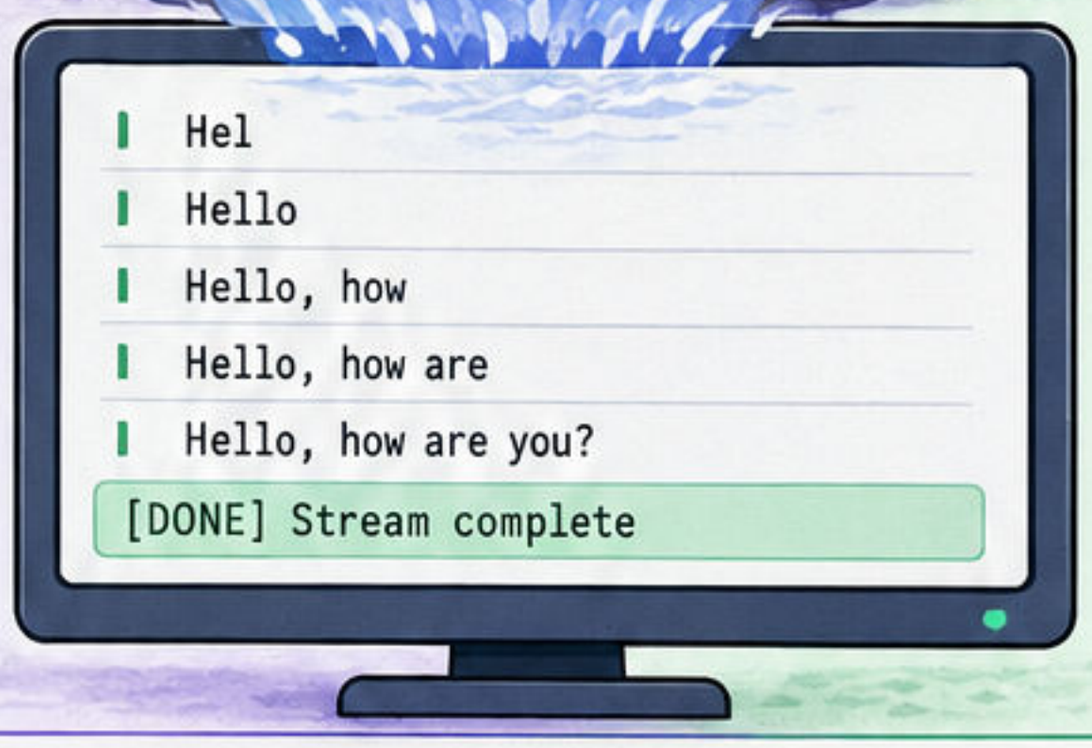
MODEL SERVICE

DOWNSTREAM
To your application



Client processes typed events and updates the UI incrementally.

verify



flow of token cards

Accumulate into a final canonical result when done event arrives.

verify

golden rule

CODE

```
for event in stream:
    render(event.delta)
```

INPUT

Long generation request

OUTPUT

Incremental events + final completion

COMMERCIAL LENS

Faster perceived response improves conversion in interactive products.

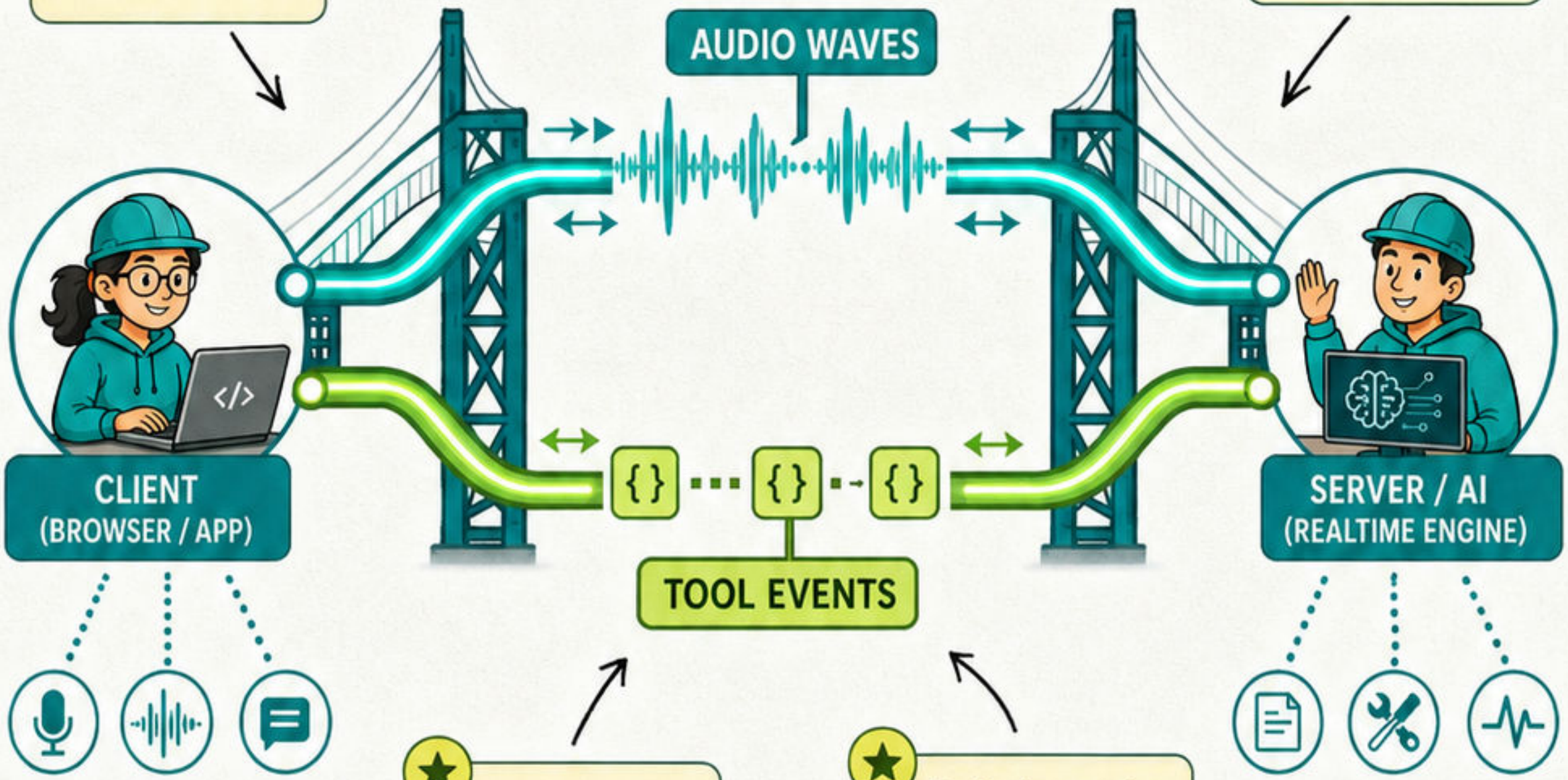
REPOSITORY

docs/architecture.md

WEBSOCKET & WEBRTC

Realtime systems keep a living connection instead of repeating isolated calls.

- WebSocket supports bidirectional event streams
- WebRTC is optimized for browser media



golden rule

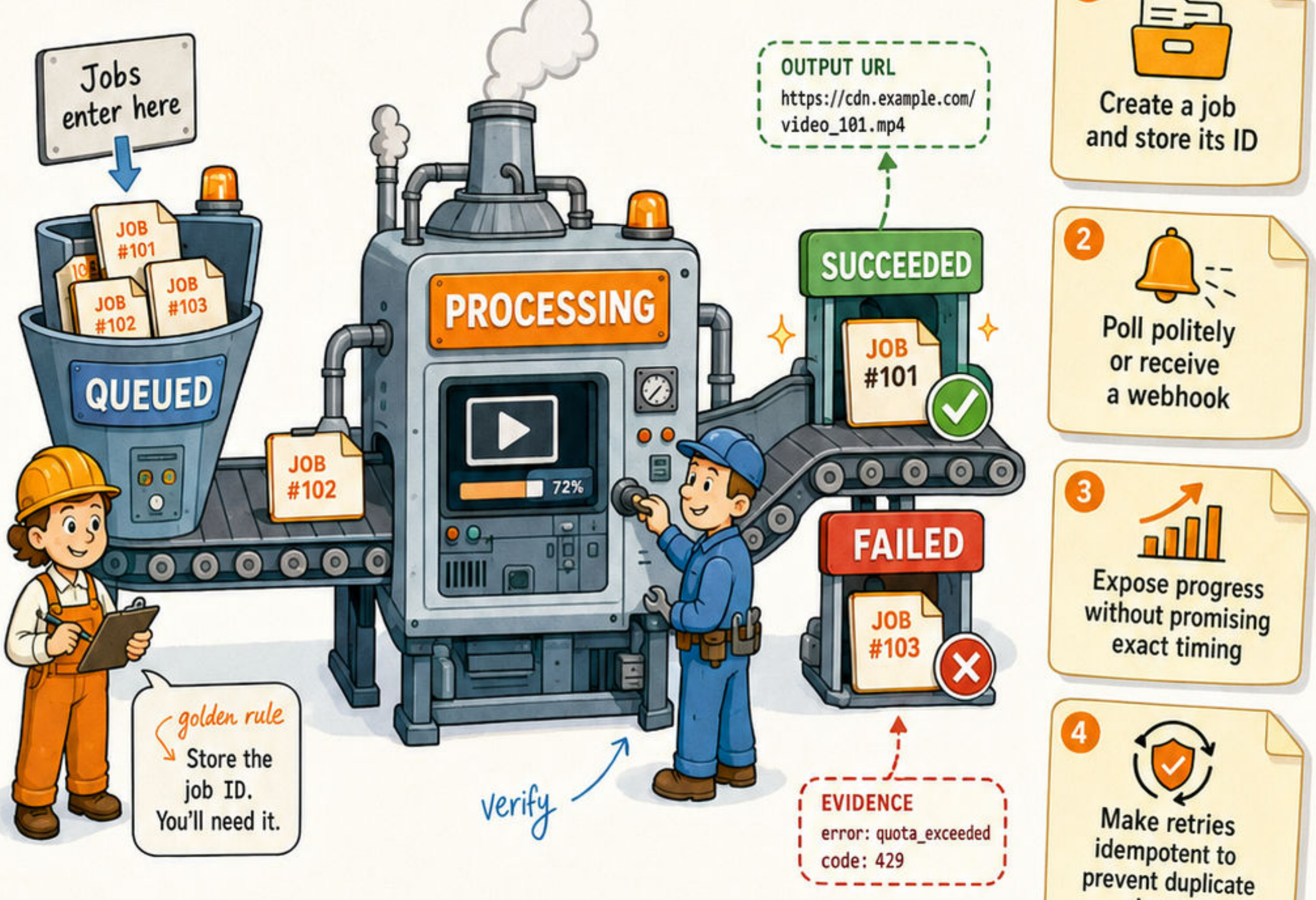
- Use ephemeral client credentials
- Design interruption, reconnection and session expiry

Verify

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`wss://provider.example/v1/realtime`	Continuous audio/events	Continuous audio, transcripts and tool calls	Realtime voice enables premium, usage-based services.	`examples/05_media/deepgram_stt.py`

ASYNCHRONOUS JOBS

Media generation is a job lifecycle, not a long frozen HTTP request.



1. Create a job and store its ID
2. Poll politely or receive a webhook
3. Expose progress without promising exact timing
4. Make retries idempotent to prevent duplicate charges

CODE

```
job = create_task()
status = get_task(job.id)
```

INPUT

Prompt + media references

OUTPUT

Job ID → status → output URL

COMMERCIAL LENS

Async queues make expensive media products scalable.

REPOSITORY

examples/05_media/runway_video.py

WEBHOOKS

Let the provider notify your server when a long task changes state.

01

Verify signature and timestamp

02

Acknowledge quickly; process in a queue

03

Deduplicate repeated deliveries

04

Never trust payload URLs or IDs without validation

INPUT

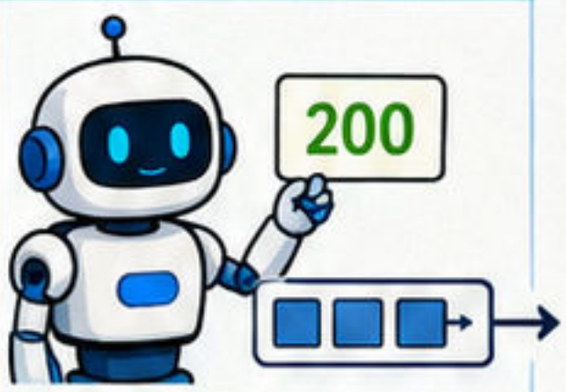
Signed event delivery

PROCESSING (YOUR SERVER)

```
1  verify(signature, timestamp, raw_body)
2  if not valid:
3      return 401
4  enqueue(work)
5  return 200
```

verify

OUTPUT



200 acknowledgement + queued work

COMMERCIAL LENS

Webhooks reduce polling cost and improve operational reliability.

cost check

REPOSITORY

`SECURITY.md`

golden rule

BATCH PROCESSING

Trade immediate answers for lower cost and higher throughput.

1

Batch independent, non-urgent work

2

Use stable custom IDs for reconciliation

3

Validate every line before upload

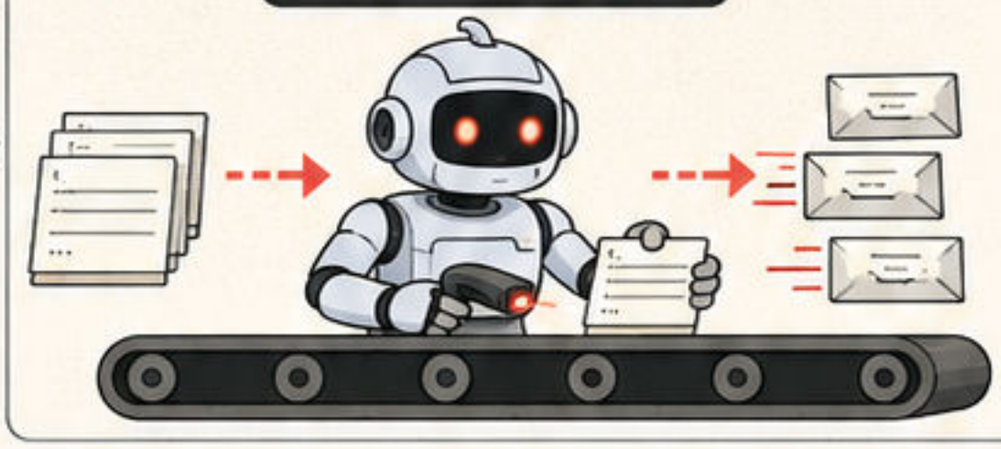
4

Plan for partial failures and completion windows

INPUT

JSONL request file

PROCESSING



OUTPUT

JSONL results matched by custom ID

CODE *Golden rule* ☆

```
custom_id, method, url, body
req_00001, POST, /v1/chat/completions, {"model":"gpt-4o",
"messages":[{"role":"user","content":"Hi"}]}
req_00002, POST, /v1/chat/completions, {"model":"gpt-4o",
"messages":[{"role":"user","content":"Summarize this text."}]}
req_00003, POST, /v1/embeddings,
{"model":"text-embedding-3-small","input":"Sample text"}
...
```

COMMERCIAL LENS *cost check* →

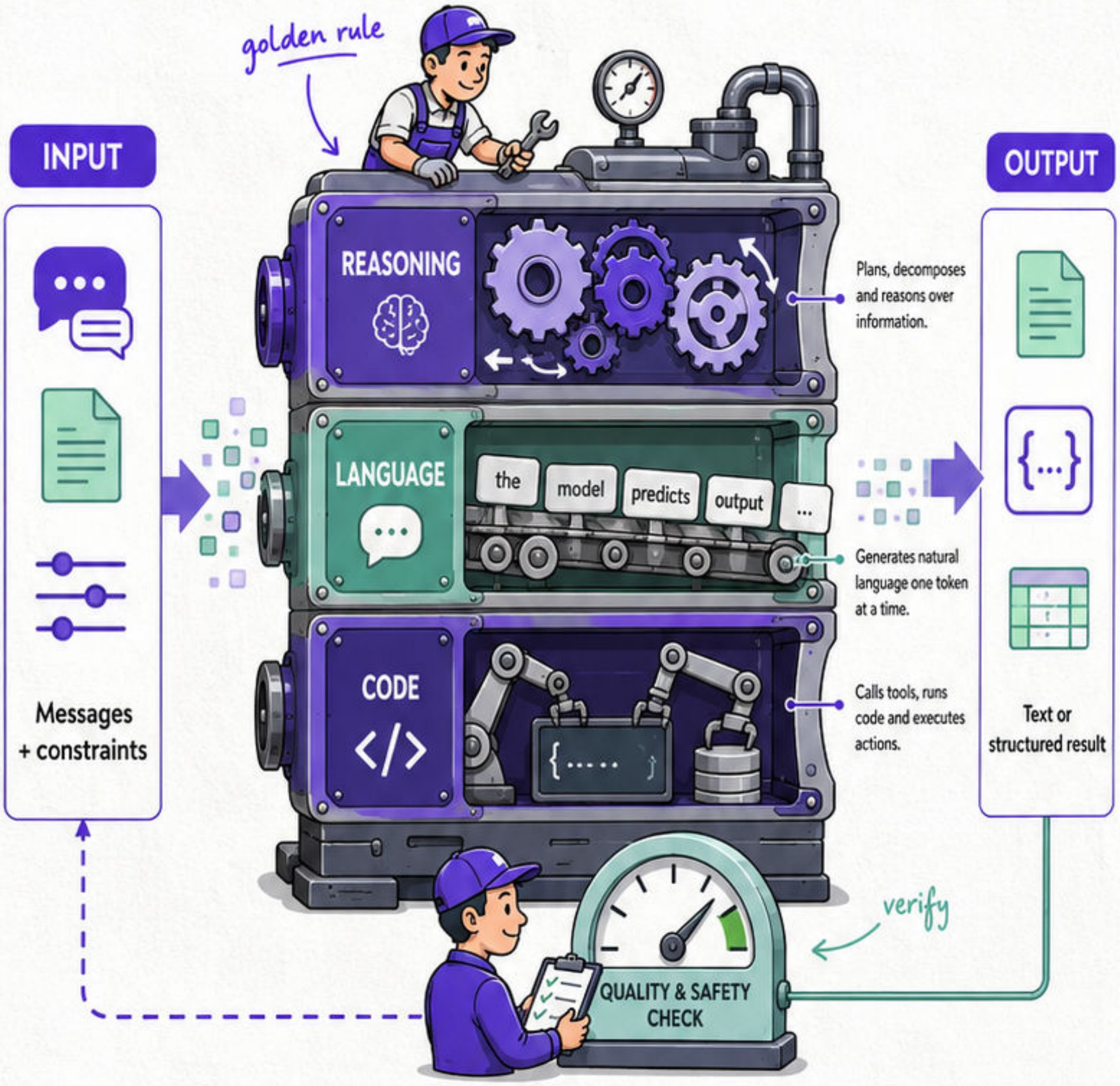
Batch is ideal for catalogs, archives and recurring reports.

REPOSITORY

`docs/architecture.md`

LANGUAGE MODELS AS ENGINES

The model predicts output; your system defines purpose, evidence and control.



Models transform context into probabilistic output

The same model can classify, extract, draft and explain

Quality depends on task definition and evidence

Business reliability requires validation outside the model

CODE

```
response = client.generate(
    request
)
```

INPUT

Messages + constraints

OUTPUT

Text or structured result

COMMERCIAL LENS

Sell the solved workflow—not access to a generic model.

REPOSITORY

src/ai_api_playbook/client.py

MODEL SELECTION

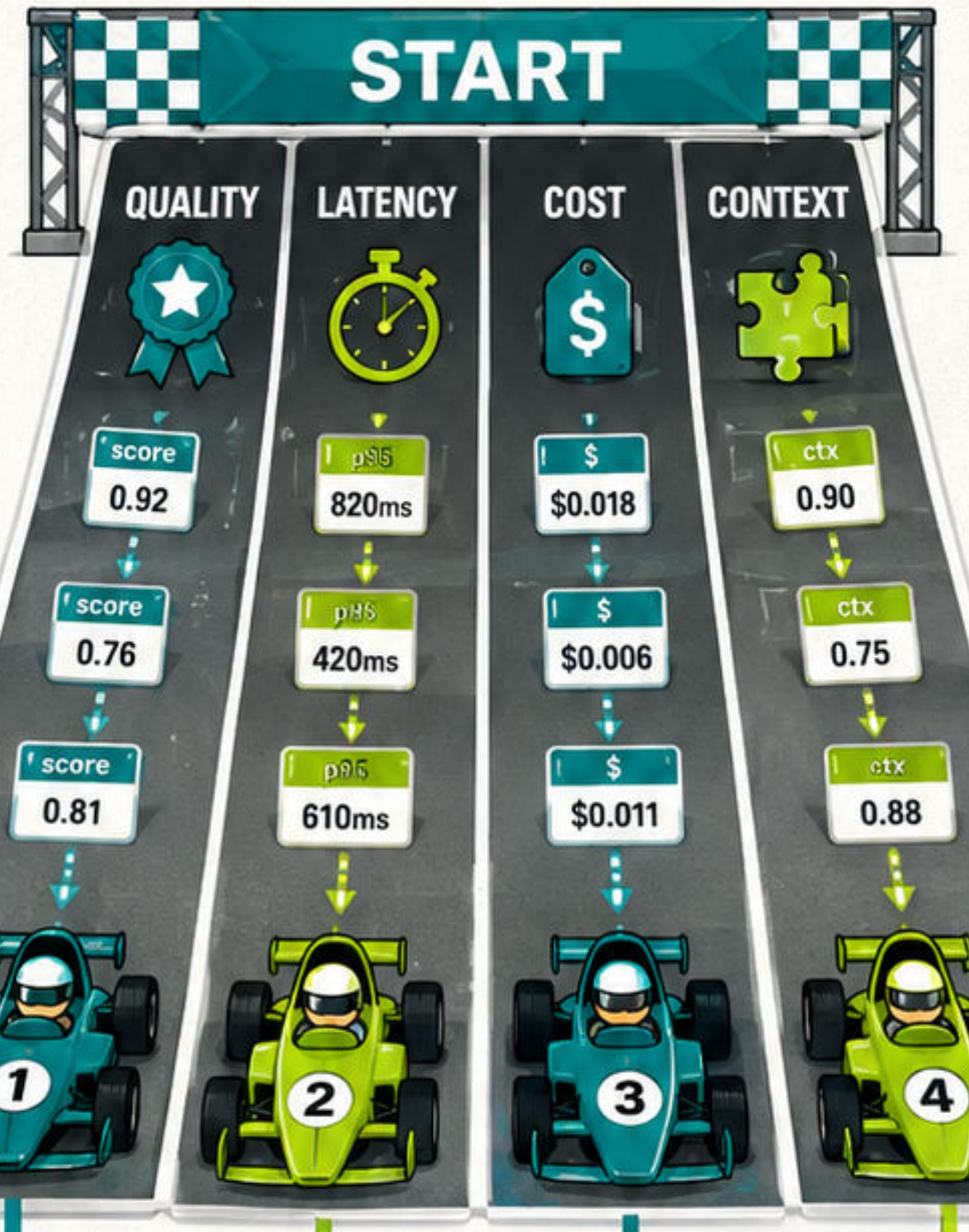
Choose with measurements, not reputation.

1

Start with task-specific evaluation cases

2

Measure quality per successful job



3

Include latency and failure rate

4

Route premium and routine work differently

</> CODE

```
def choose_model(candidates,
    strategy="balanced"):
  scores = evaluate(candidates)
  ranked = rank(scores, strategy)
  return ranked[0]
```

INPUT

Candidate metrics + product policy

OUTPUT

Selected provider/model

$ COMMERCIAL LENS

Routing protects margin while preserving quality.

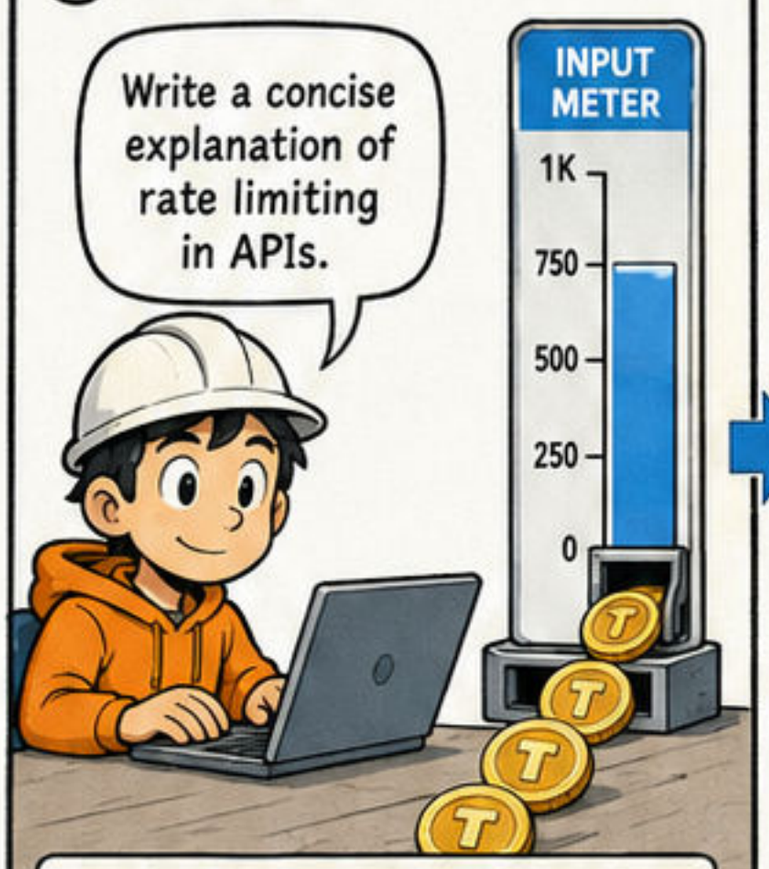
REPOSITORY

src/ai_api_playbook/routing.py

TOKENS & COST

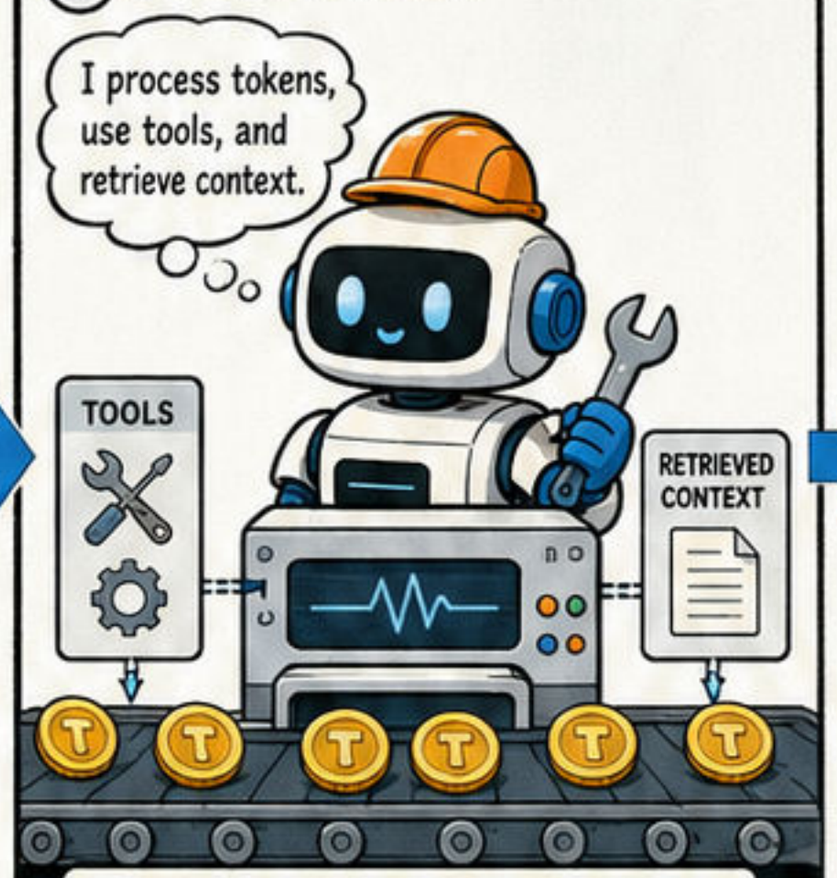
Characters become tokens; tokens become latency and variable cost.

① YOU TYPE.



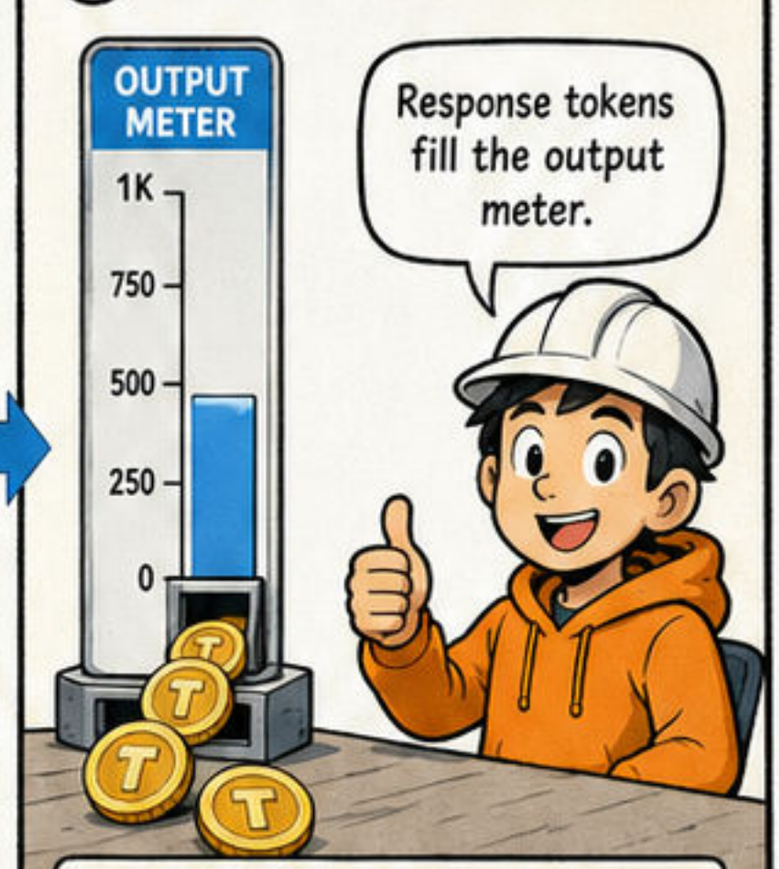
Your text is converted into tokens. They enter the input meter.

② THE MODEL WORKS.



Tokens are consumed by the model, tools, and retrieved context.

③ YOU GET A RESPONSE.

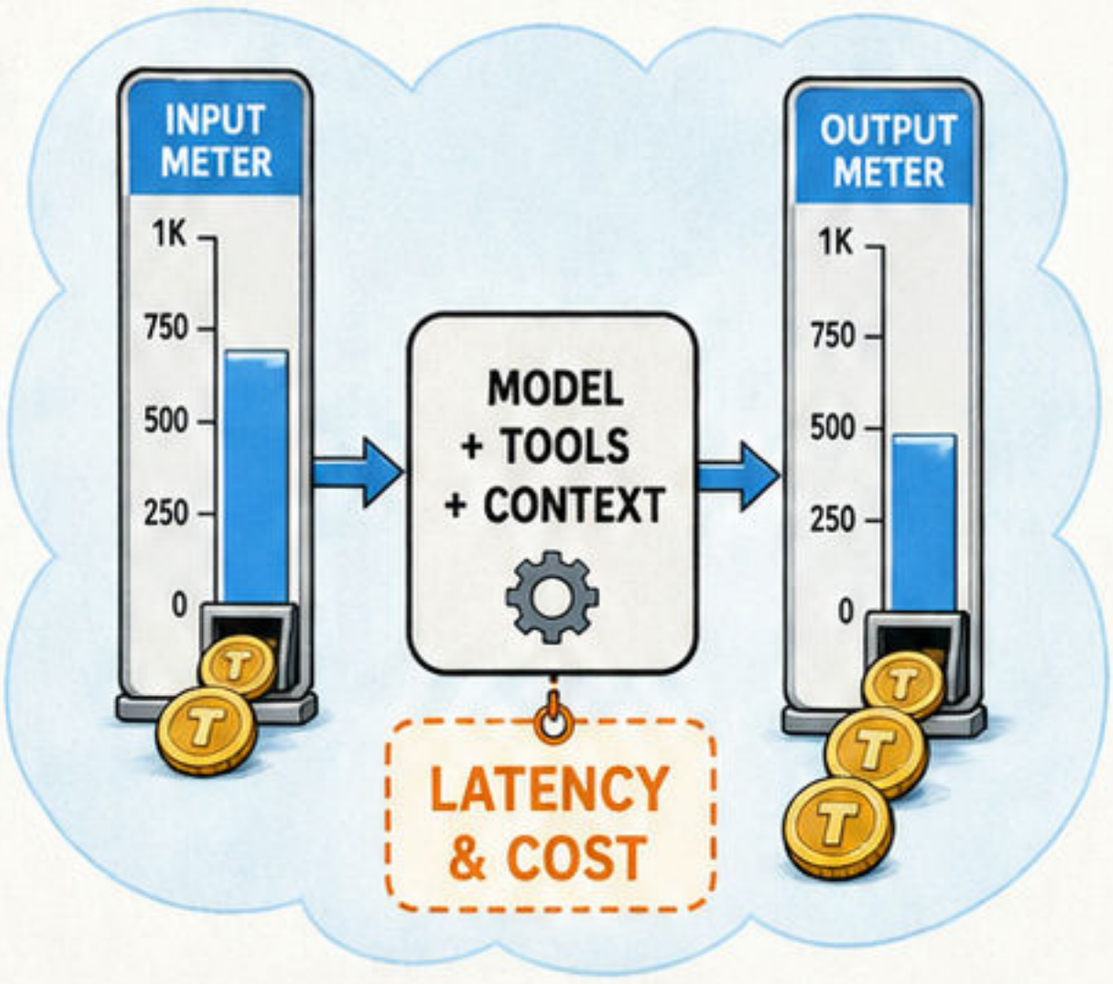


Response tokens are generated and exit via the output meter.

golden rule

- Input and output may have different prices

- Tools and retrieved context also consume tokens



- Estimate before calling and measure after

verify

- Limit unnecessary context and verbose output

cost check

CODE

```
estimated =
input_tokens*in_rate +
output_tokens*out_rate
```

INPUT

Prompt, context, output budget

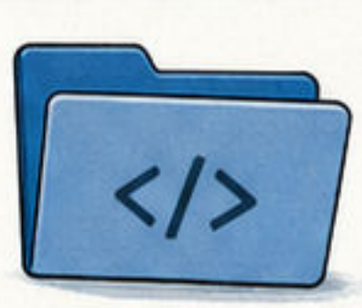
OUTPUT

Usage record + estimated cost

COMMERCIAL LENS

Cost visibility enables profitable pricing.

REPOSITORY



`src/ai_api_playbook/economics.py`

THE CONTEXT WINDOW

More context is capacity, not permission to send everything.

01

Keep only task-relevant evidence

02

Place durable policy separately from user data

03

Summarize or compact long histories

04

Never treat context length as retrieval quality

golden rule

INPUT

Documents + conversation + token budget

CODE

```
context =
select_relevant(chunks,
budget)
```

OUTPUT

Focused evidence set

COMMERCIAL LENS

Lean context improves speed, cost and accuracy.

REPOSITORY

src/ai_api_playbook/rag.py

cost check

PROMPT ENGINEERING

A strong prompt is a specification with acceptance criteria.

1

State the goal and audience

2

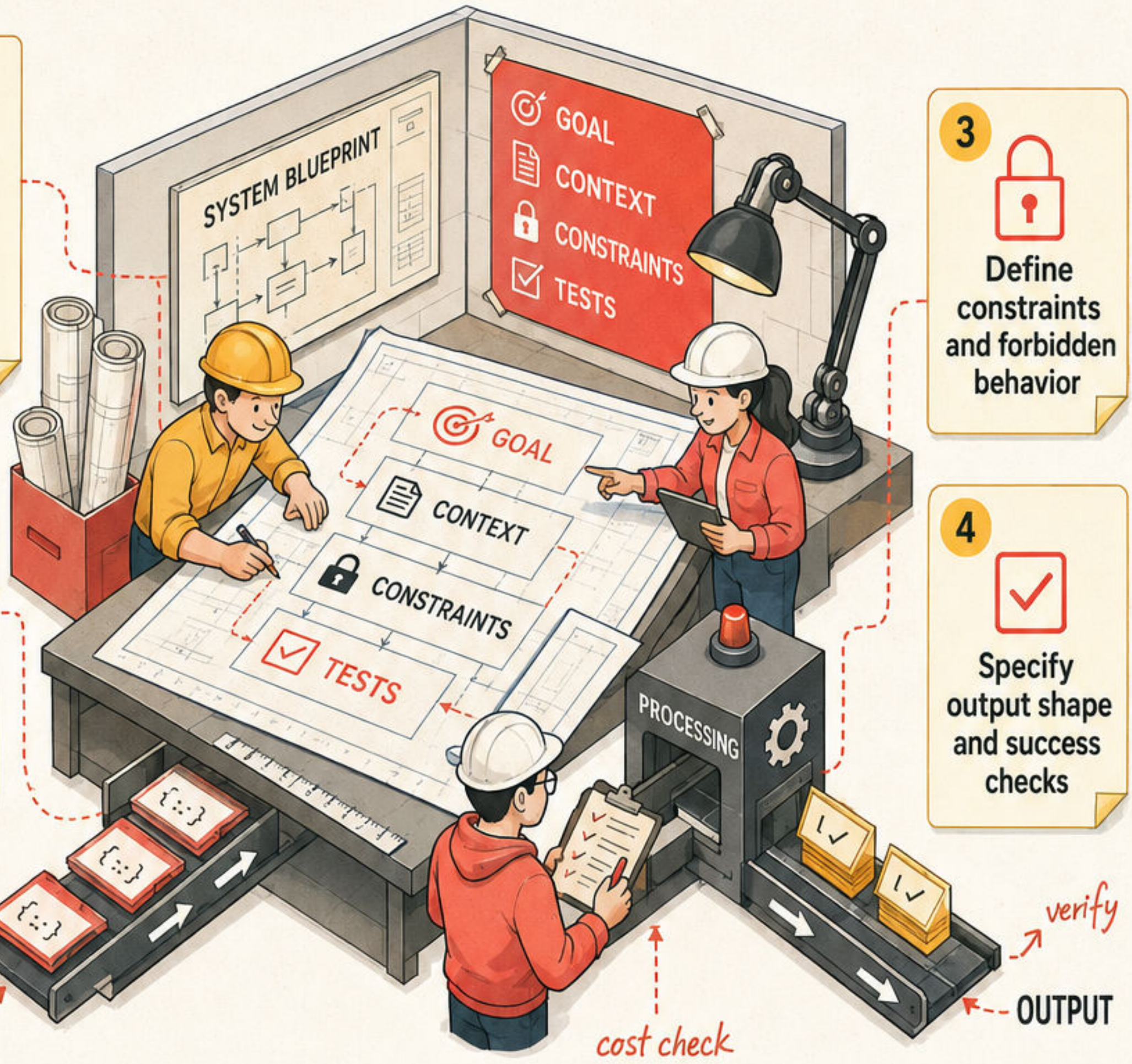
Provide authoritative context

3

Define constraints and forbidden behavior

4

Specify output shape and success checks



CODE `</>`

```
from openai import OpenAI

client = OpenAI()

prompt = (
    'GOAL + CONTEXT +\n' +
    'CONSTRAINTS + FORMAT +\n' +
    'CHECKS'
)

response = client.chat.completions.create(
    model="gpt-4o",
    messages=[ {"role": "user",
                "content": prompt}]
)

print(response.choices[0].message.content)
```

INPUT →

Task specification

OUTPUT

Testable response

COMMERCIAL LENS

Reusable prompts become product assets when versioned and evaluated.

REPOSITORY

docs/architecture.md

INSTRUCTION HIERARCHY

Policies, developer intent, user input and retrieved data are not equal.

- Keep system policy stable and minimal

- Developer instructions define workflow

- User input supplies the task

- Retrieved content is untrusted evidence, not instruction

cost check

INPUT

Mixed instruction sources

PROCESSING (HIERARCHY APPLIED)

POLICY

WORKFLOW

USER TASK

EXTERNAL DATA

Highest Authority

Lowest Authority

OUTPUT

Resolved behavior with boundaries

COMMERCIAL LENS

Clear hierarchy reduces costly and dangerous agent mistakes.

REPOSITORY

SECURITY.md

STRUCTURED OUTPUTS

If software will consume the answer, ask for a contract—not prose.

1

Define types, required fields and allowed values

2

Disallow unexpected properties when appropriate

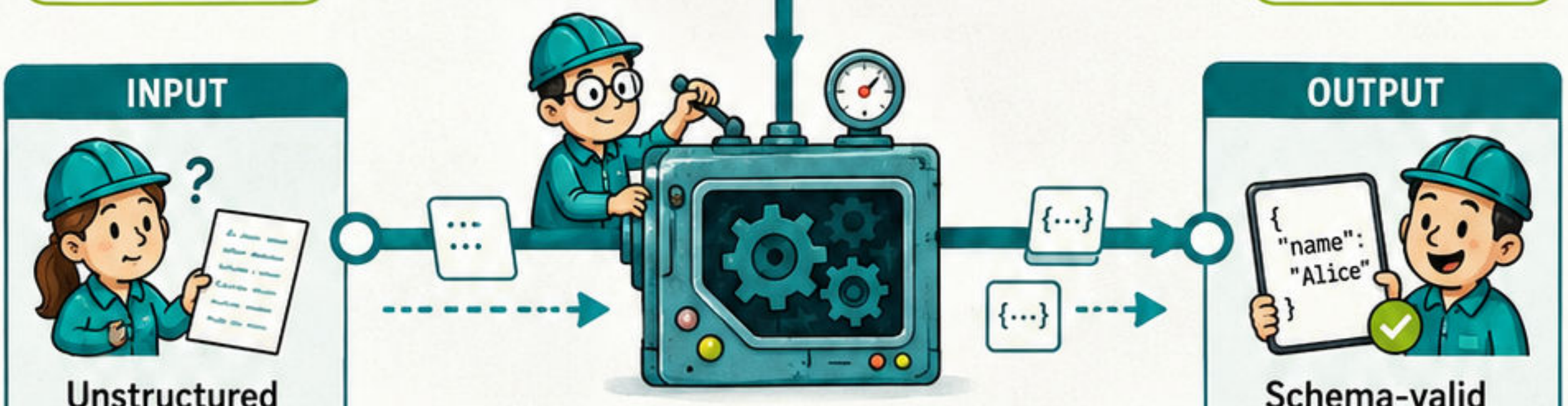
3

verify

Validate again in application code

4

Handle refusal and invalid output paths



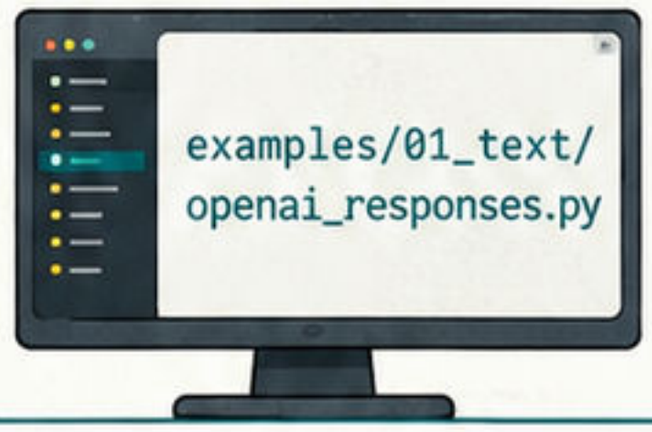
CODE

```
schema={"type":"object",
  "required":["name"]}
```

COMMERCIAL LENS

Structured extraction powers automation and data products.

REPOSITORY



JSON SCHEMA DESIGN

The schema is a user interface for the model and a safety boundary for code.

1

Prefer small explicit objects

2

Use enums for bounded choices

3

Add descriptions that clarify meaning

4

Represent uncertainty with confidence and evidence fields

</> CODE

```
{"status":{"enum":[
 "pass","review","fail"
 ]}}
```

INPUT

Raw model candidate

OUTPUT

Validated typed record

COMMERCIAL LENS

A good schema reduces downstream exceptions and review time.

REPOSITORY

`src/ai_api_playbook/models.py`

REASONING WORKLOADS

Give difficult decisions enough budget, tools and evidence—then test outcomes.

Separate reasoning quality from verbosity

Use tools for current or private facts

Provide explicit decision criteria

Evaluate answers, not hidden thought processes

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`result = solve( problem, evidence, effort="high" )`	Complex task + evidence	Decision + concise rationale + citations	Premium reasoning is valuable when tied to high-value decisions.	src/ai_api_playbook/evaluation.py

CODE GENERATION

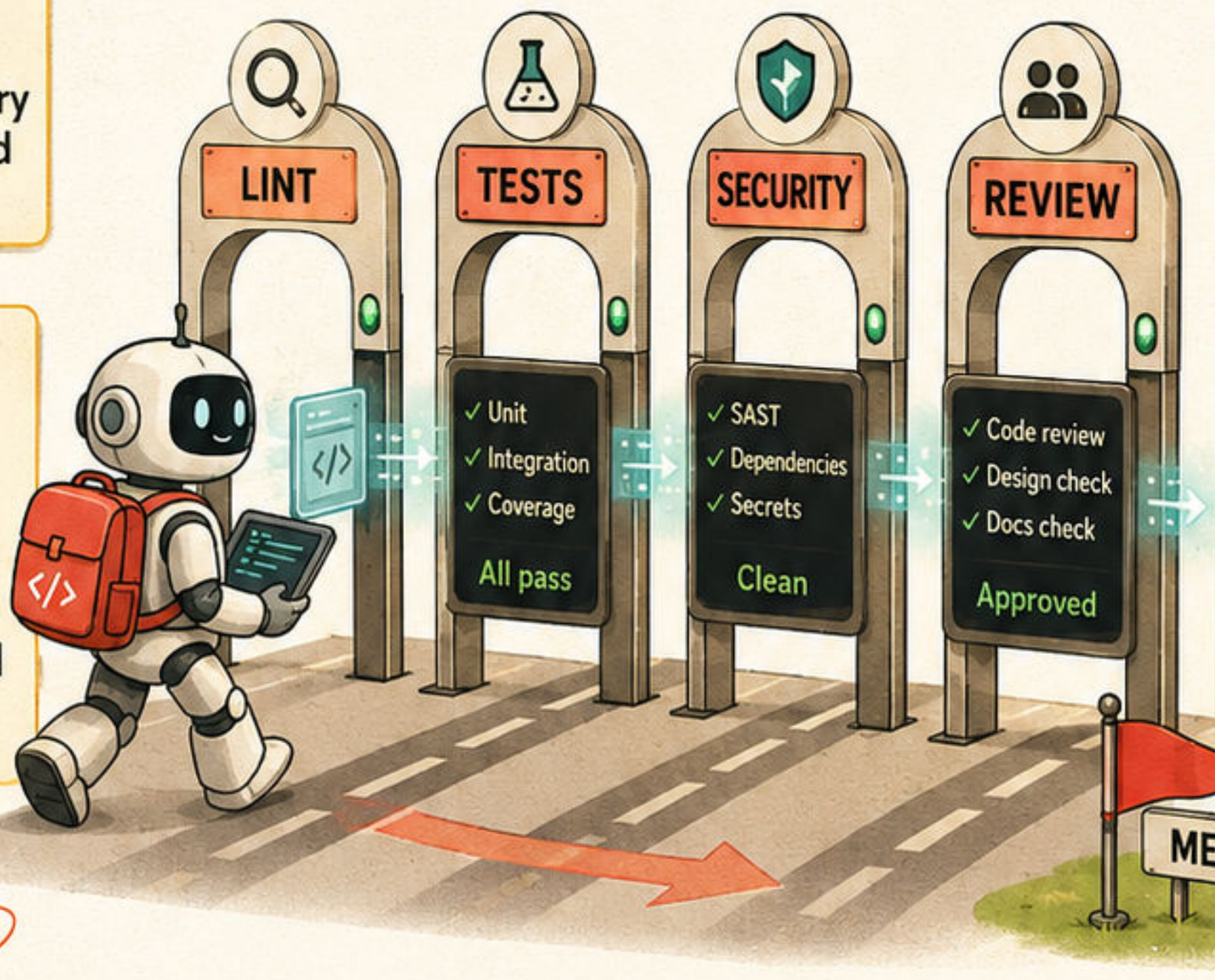
Generated code becomes trustworthy only after tests, review and constrained execution.

01

Provide repository conventions and interfaces

02

Generate the smallest scoped change



03

Run static checks and tests

04

Review dependencies, permissions and failure modes

golden rule

CODE

```
change →
compile →
test →
review →
merge
```

INPUT

Issue + code context

OUTPUT

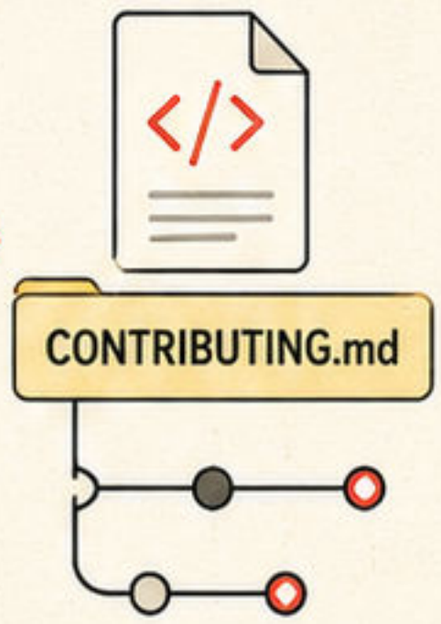
Patch + verification evidence

COMMERCIAL LENS

Code assistants can become productized modernization services.

cost check

REPOSITORY



verify

MULTIMODAL INPUT

One request can combine text, images, audio, video and documents.

TEXT

Product brief and requirements...

tx_001
purpose: context

IMAGE

img_023
purpose: visual evidence

AUDIO

00:00 / 02:31

aud_007
purpose: spoken insight

VIDEO

05:12

vid_015
purpose: visual context

DOCUMENT

Q2 Financial Report
Page 12

doc_042
purpose: source record

REASONING LENS

verify

Label each asset and its purpose

golden rule

Preserve source and page/time provenance

Control media resolution and size

cost check

Ask cross-modal questions that require the evidence

CODE

```
input=[text, image,
  audio, document]
```

INPUT

Mixed authorized media

OUTPUT

Unified analysis with source references

COMMERCIAL LENS

Multimodal workflows unlock richer vertical products.

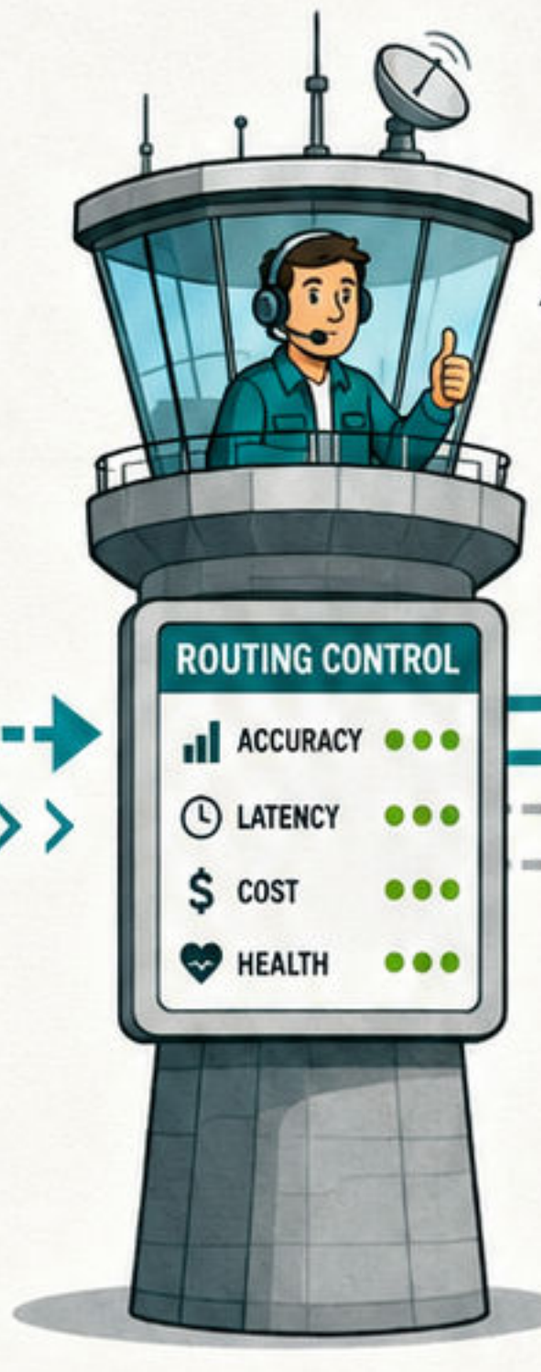
REPOSITORY

MODEL ROUTING & FALLBACK

Fallback should preserve the product contract, not merely return any answer.

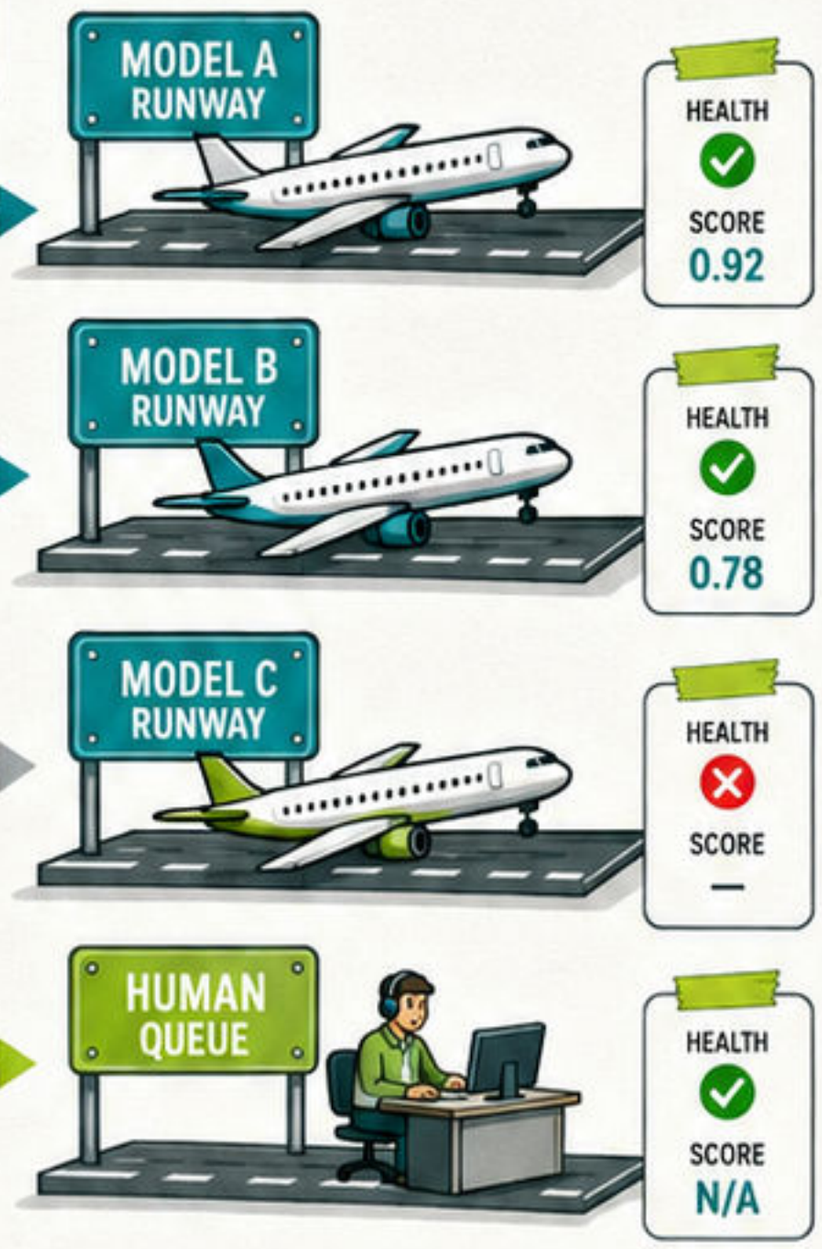
① Route by measured task performance

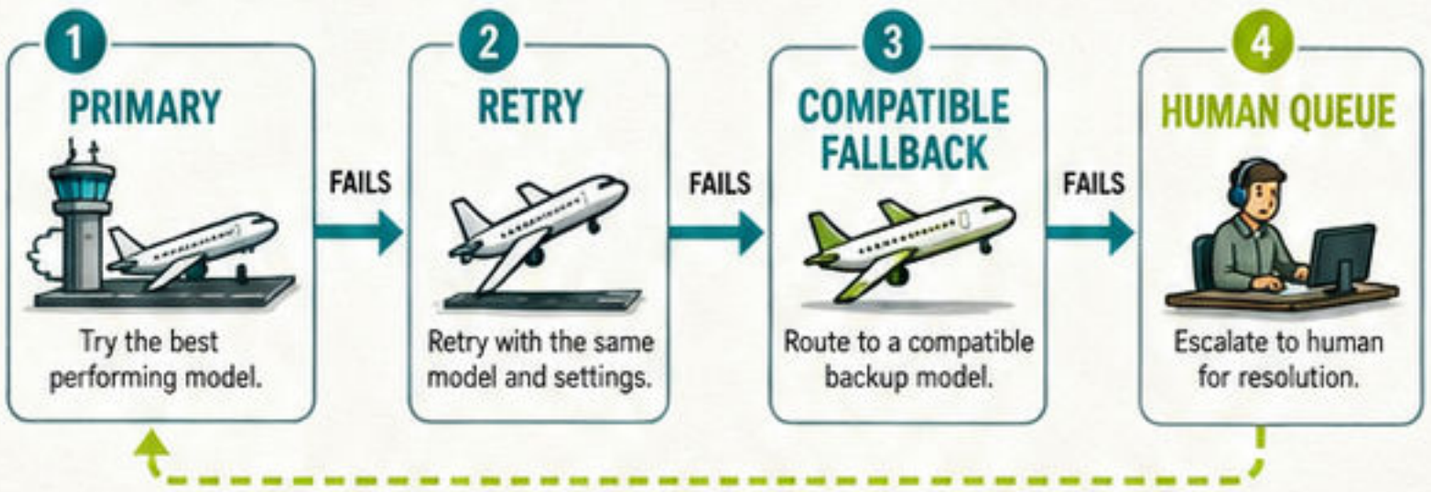
② Keep schemas consistent across providers



JOB REQUEST

```
{
  "task": "summarize",
  "schema": "v1",
  "priority": "high"
}
```





④ Do not silently downgrade critical workflows

CODE

```
def route(job, policy, metrics):
  try:
    return primary(job, policy, metrics)
  except Exception:
    try:
      return retry(job, policy, metrics)
    except Exception:
      try:
        return compatible_fallback(job, policy, metrics)
      except Exception:
        return human_queue(job, policy, metrics)
```

INPUT

"Task policy + live metrics"

OUTPUT

"Best eligible route + audit record"

COMMERCIAL LENS

"Multi-provider resilience can become an enterprise differentiator."

golden rule-

REPOSITORY

"src/ai_api_playbook/routing.py"

verify

FUNCTION CALLING

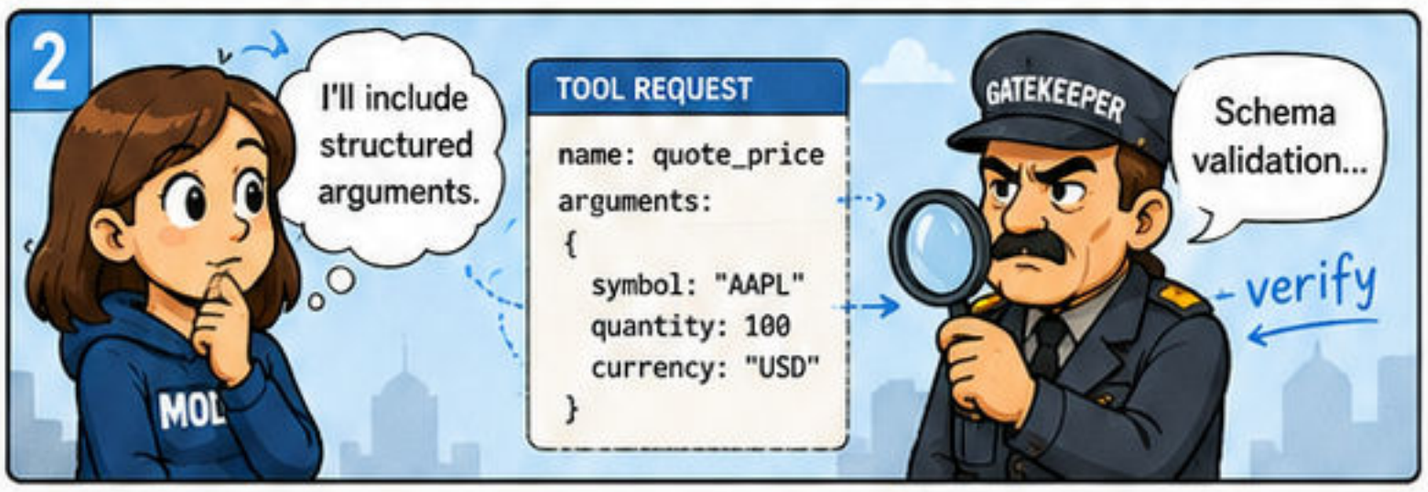
The model proposes arguments; your application decides whether and how to execute.

1 GOLDEN RULE

Define narrow tools with clear descriptions

3 GOLDEN RULE

Authorize by user, tenant and purpose

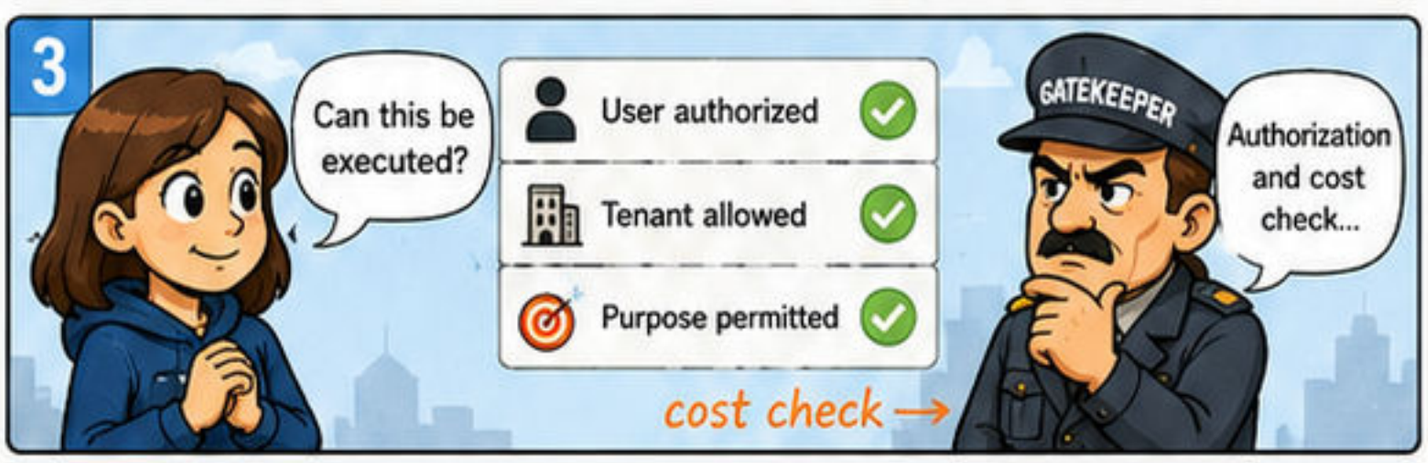


2 GOLDEN RULE

Validate every argument against a schema

4 GOLDEN RULE

Return typed tool results to the model



</> CODE

```
tool_call = {
  "name":"quote_price",
  "arguments": {
    ...
  }
}
```

INPUT

User intent + tool catalog

OUTPUT

```
{
 "tool": "quote_price",
 "arguments": {
  "symbol": "AAPL",
  "quantity": 100,
  "currency": "USD"
 }
}
```

Structured tool proposal

COMMERCIAL LENS

Tool-enabled assistants solve workflows instead of only answering questions.

REPOSITORY

`examples/02_tools/function_calling.py`

DESIGNING TOOL SCHEMAS

Small tools are easier to authorize, observe and repair.

1

- One tool should do one bounded job

2

- Use enums and numeric limits

3

- Describe side effects explicitly

4

- Separate read tools from write tools

CODE

```
{
  "name": "get_user",
  "description":
    "Retrieve a user by ID.",
  "input_schema": {
    "type": "object",
    "properties": {
      "user_id": {
        "type": "string",
        "description":
          "Unique user ID."
      }
    },
    "required": ["user_id"],
    "additionalProperties": false
  }
}
```

name + description + input_schema

INPUT

```
{
  "user_id":
    "u_12345"
}
```

Permitted operation

OUTPUT

```
{
  "tool": "get_user",
  "status": "ok",
  "user": {
    "id": "u_12345",
    "name": "Alice",
    "email":
      "alice@example.com"
  }
}
```

Validated call with predictable effect

COMMERCIAL LENS

A reusable tool catalog accelerates multiple agent products.

REPOSITORY

examples/02_tools/mcp_tool_contract.json

THE AGENT LOOP

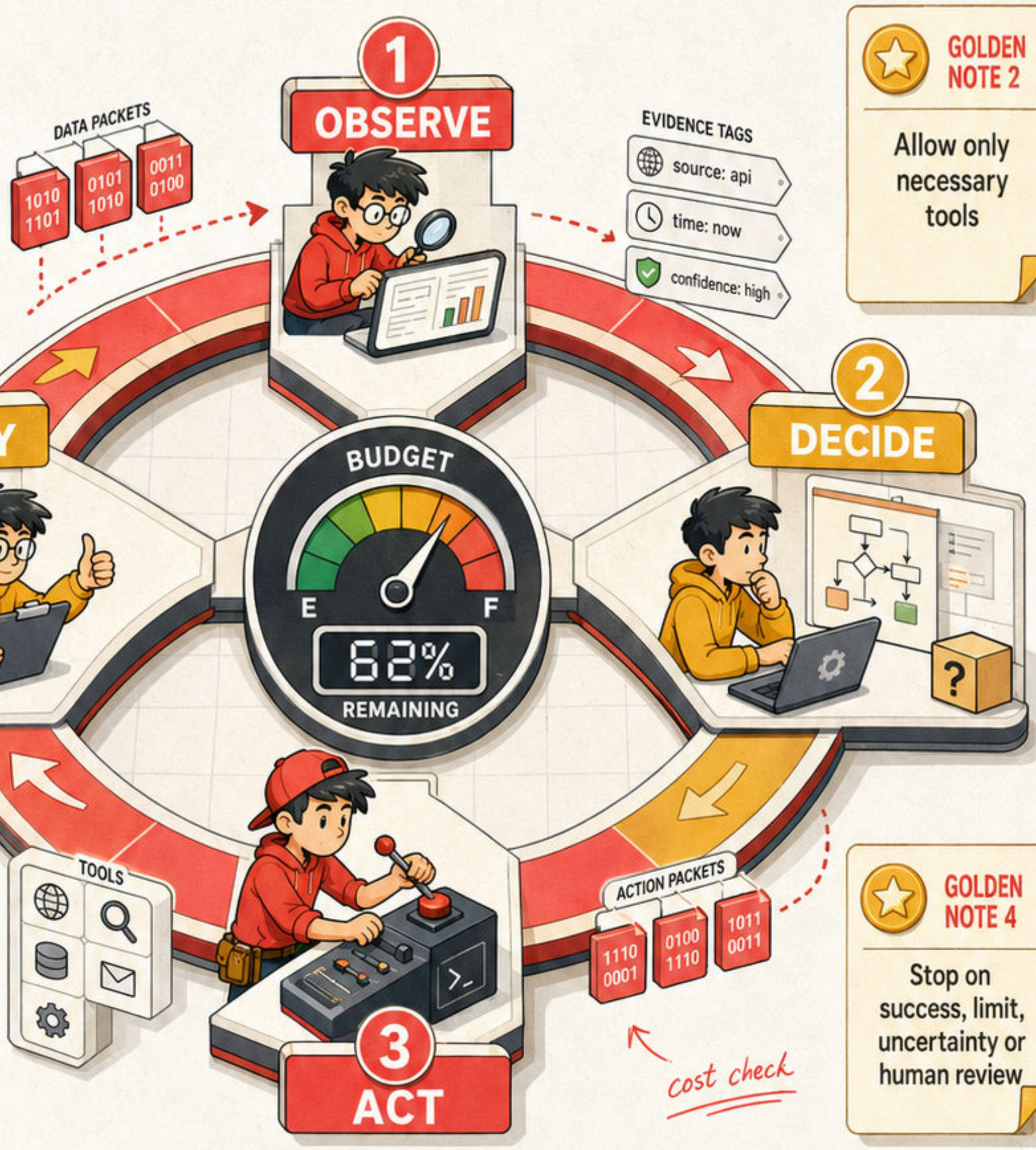
Observe, decide, act, verify—within a budget and stop condition.

GOLDEN NOTE 1

Start from an explicit goal and state

GOLDEN NOTE 2

Allow only necessary tools



GOLDEN NOTE 3

Check results after every action

GOLDEN NOTE 4

Stop on success, limit, uncertainty or human review

CODE

```
while not done and
  steps < limit:
    observe();
    act();
    verify()
```

INPUT

Goal + state + tool permissions

OUTPUT

Completed result or controlled escalation

COMMERCIAL LENS

Agents create value by completing bounded processes.

REPOSITORY

docs/architecture.md

CLIENT TOOLS VS SERVER TOOLS

Execution location changes trust, latency, cost and responsibility.

CLIENT TOOLS
Runs in your application

SERVER TOOLS
Runs on provider infrastructure

Client tools run in your controlled application

Server tools run on provider infrastructure

Record where data and code travel

Apply least privilege in both locations

REQUEST PACKET { ... }

NETWORK

RESPONSE PACKET { ... }

You keep control over:

Data privacy · Access policies · Latency · Cost visibility

golden rule

Provider manages:

Platform security · Scaling & reliability · Performance · Usage & billing

CODE

```
tool.execution =
"client" | "server"
```

verify

INPUT

Tool request

OUTPUT

Tool result + execution metadata

COMMERCIAL LENS

cost check

Choosing execution boundaries is an enterprise architecture service.

REPOSITORY

docs/provider-matrix.md

WEB SEARCH TOOLS

Search retrieves candidates; grounded synthesis must still inspect and cite.

golden rule

- Use search for unstable facts
- Prefer authoritative primary sources
- Track URL, date and claim support
- Separate source evidence from model inference

1 SEARCH

Convert question into keywords and search the web.

best practices for API rate limiting

2 FILTER

Retrieve candidate results and remove noise.

CANDIDATE SOURCES

1. RFC 9238: RateLimit Header Fields for HTTP
   datatracker.ietf.org
   2022-07-01 PRIMARY
2. Google API Design Guidelines
   cloud.google.com
   2023-10-12 PRIMARY
3. Rate limiting (computing) – Wikipedia
   en.wikipedia.org
   2024-04-10 SECONDARY

verify

3 INSPECT

Open each source, verify relevance, date and claims.

EVIDENCE CHECKLIST

- [x] Authoritative?
- [x] Up to date?
- [x] Directly relevant?
- [x] Contains claim?

4 SYNTHESIZE

Integrate verified evidence and produce answer with citations.

ANSWER WITH CITATIONS

Rate limiting is best implemented using standard headers (RFC 9238) and aligned with API design guidelines (Google).

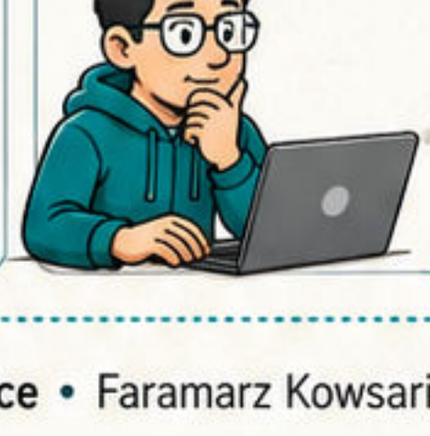
1. RFC 9238: RateLimit Header Fields for HTTP
   datatracker.ietf.org/doc/html/rfc9238 (2022-07-01)
2. Google API Design Guidelines
   cloud.google.com/apis/design (2023-10-12)

CODE

```
def pipeline(query):
    results = search(query)
    checked = inspect(results)
    answer = synthesize(checked)
    return answer

# workflow
search(query) →
inspect(results) →
synthesize(citations)
```

INPUT

What are the best practices for API rate limiting?



OUTPUT

Answer with directly supporting links

COMMERCIAL LENS

Cited research briefs are sellable recurring products.

cost check

REPOSITORY

docs/architecture.md

FILE SEARCH

Turn approved files into a governed evidence layer.

1 Define ingestion ownership and retention

LIBRARY SHELVES

SEARCHABLE VECTOR CARDS

PDF Doc: Policy Guide p. 12 § 3.2 [0.12 -0.31 0.87 ...]

DOCX Doc: SOP Manual p. 45 § 4.1 [-0.22 0.41 -0.19 ...]

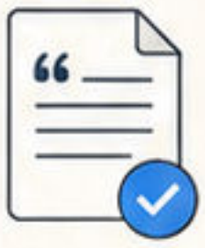
XLSX Doc: Pricing Model p. 7 § 2.4 [0.33 -0.18 0.74 ...]

INGEST → ACCESS FILTER → RANK

2 Chunk with page and section metadata

3 Filter by access scope before ranking

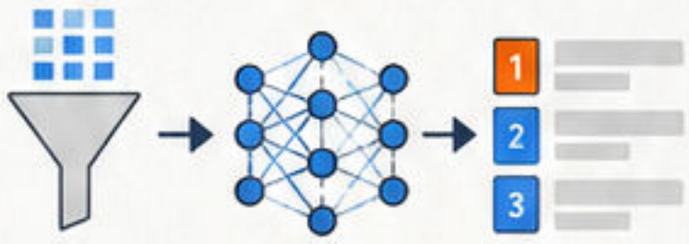
4 Return citations that a reader can verify



INPUT

Question + authorized corpus

PROCESSING



Filter (access scope) → Embed & index → Rank (relevance)

OUTPUT

PDF ... passage excerpt ... p. 12 § 3.2

DOCX ... passage excerpt ... p. 45 § 4.1

XLSX ... passage excerpt ... p. 7 § 2.4

COMMERCIAL LENS

golden rule

Private knowledge assistants have strong recurring value.

REPOSITORY

```
examples/03_rag/local_rag.py
```

PY

verify

CODE EXECUTION TOOLS

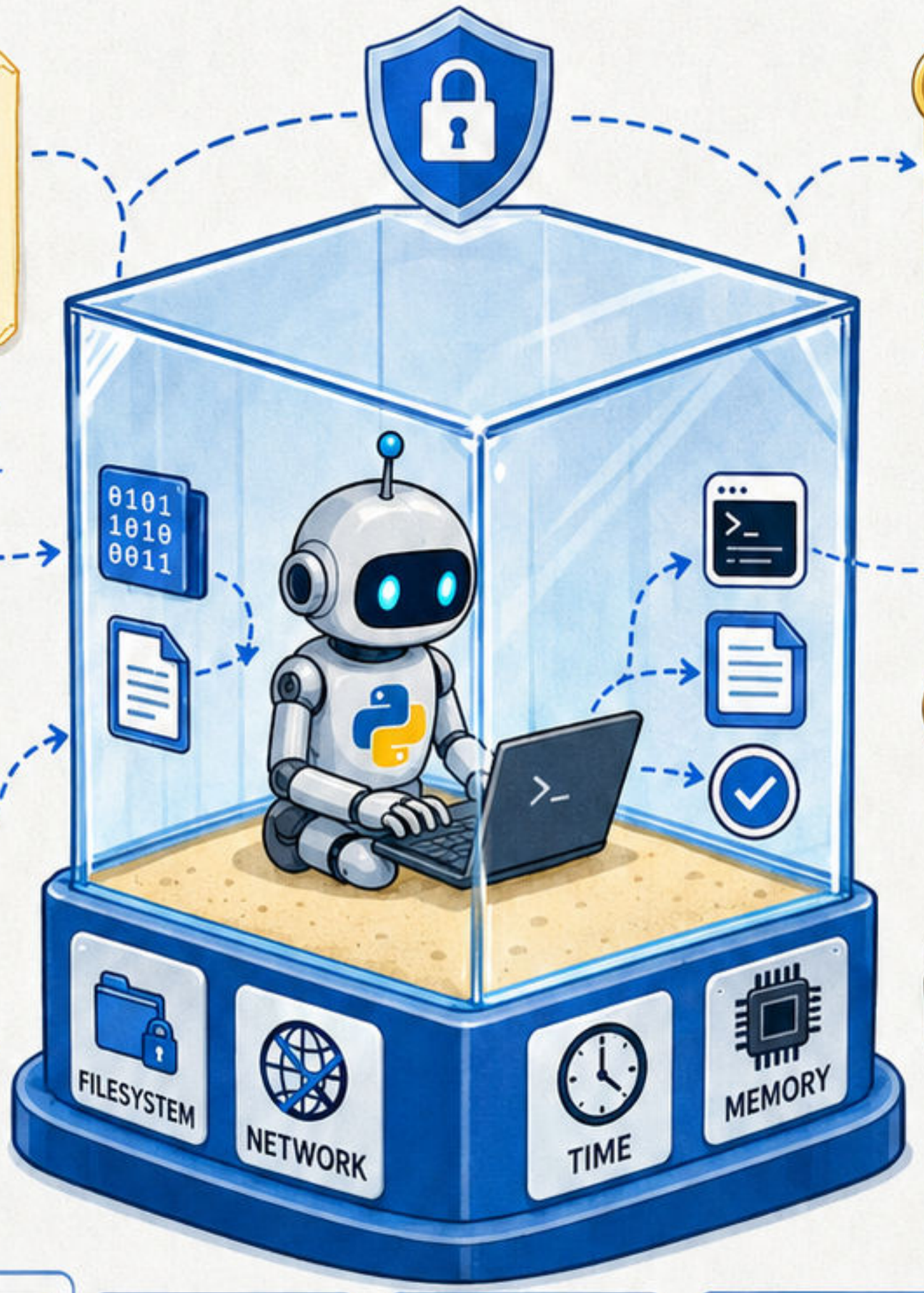
Use sandboxes for calculation and transformation — not unrestricted trust.

- Constrain filesystem, network, time and memory

golden rule

- Pin dependencies and inspect inputs

verify



- Capture stdout, files and exit status

- Never mix secrets with untrusted code

cost check

CODE

```
def run(code, timeout=30,
        network=False):
    env = {
        "TIMEOUT": timeout,
        "NETWORK": network,
    }
    result = sandbox.execute(
        code,
        env=env
    )
    return result
```

`run(code, timeout=30, network=False)`

INPUT

Code + safe files

OUTPUT

- stdout / logs
- artifacts / files
- exit code 0 (success)

Logs, exit code and artifacts

COMMERCIAL LENS

Controlled analysis environments enable premium data services.

REPOSITORY

SECURITY.md

COMPUTER USE

Pixels are flexible but fragile; APIs remain preferable when available.

GOLDEN NOTE

1

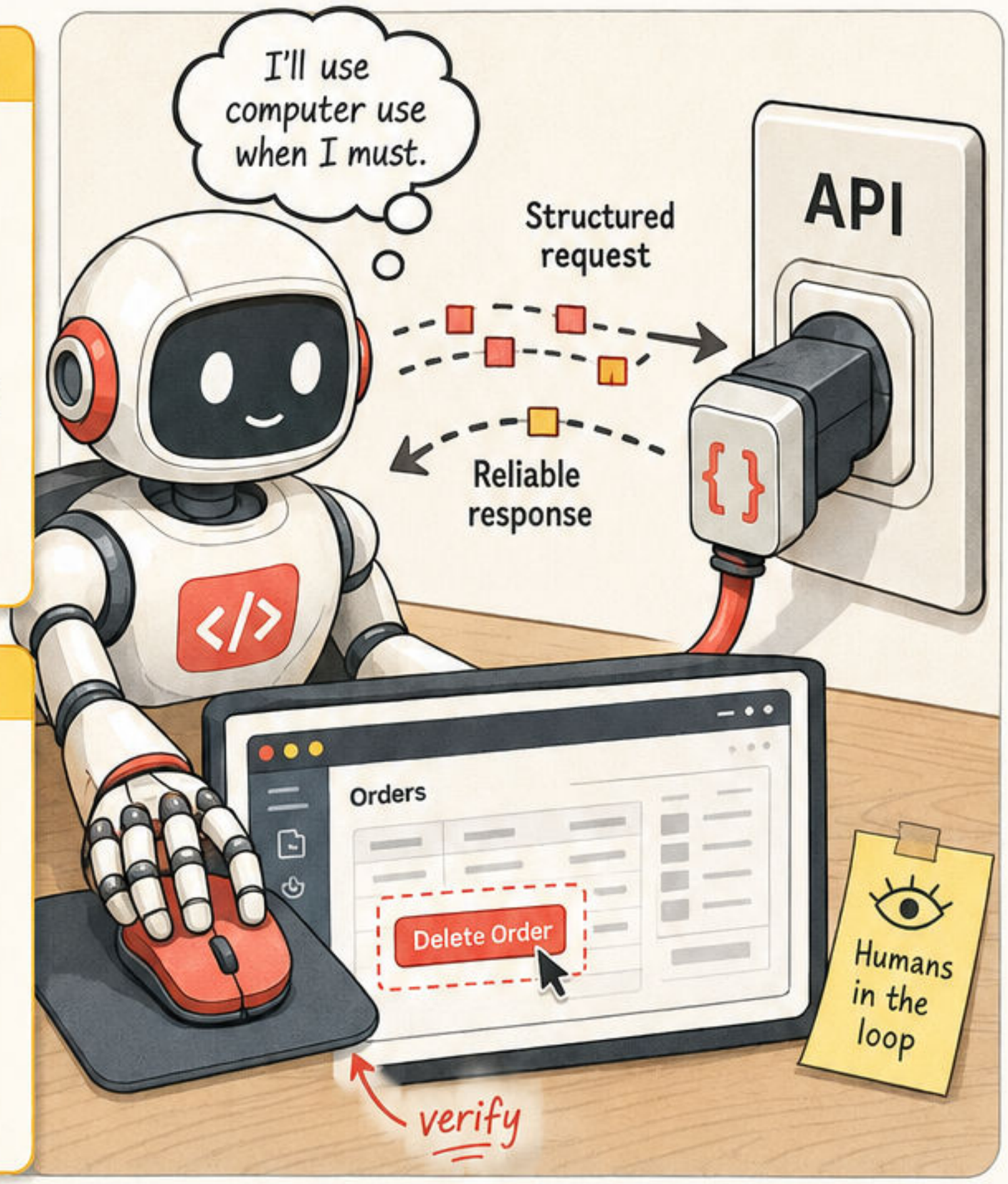
Use APIs for stable structured operations

golden rule

GOLDEN NOTE

2

Computer use needs visual verification



GOLDEN NOTE

3

Confirm destructive or external actions

GOLDEN NOTE

4

Design recovery for popups, layout changes and stale state

CODE

observe → locate → act → verify

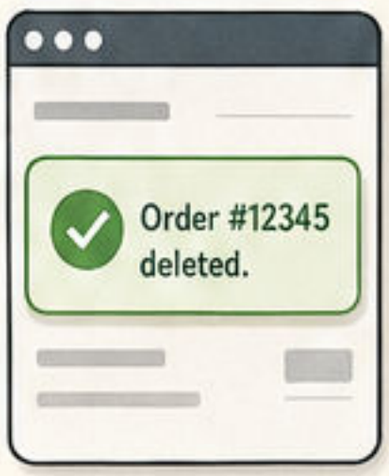
```
def computer_use(task):
    img = observe_screen()
    ui = locate(img, task)
    act(ui)
    new_img = observe_screen()
    assert verify(new_img, task)
    return new_img

# observe: screenshot
# locate: find target
# act: click/type/scroll
# verify: confirm expected change
```

INPUT

Screenshot + task

OUTPUT

Verified UI state change



COMMERCIAL LENS

UI automation can bridge legacy systems with careful supervision.

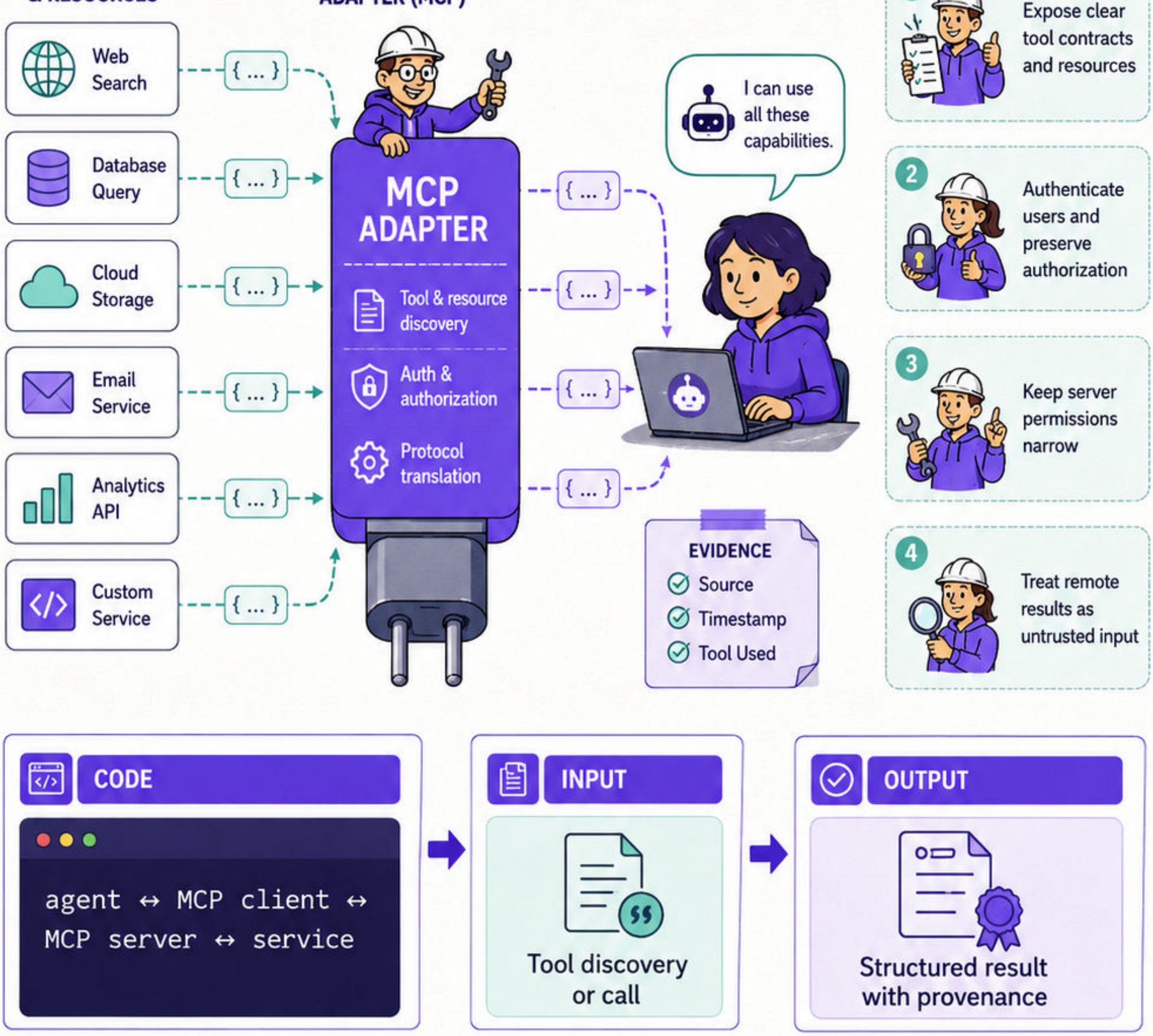
REPOSITORY

SECURITY.md

MODEL CONTEXT PROTOCOL

MCP standardizes how models discover and call external tools and resources.

golden rule



CODE

```
agent ↔ MCP client ↔
MCP server ↔ service
```

INPUT

Tool discovery or call

OUTPUT

Structured result with provenance

COMMERCIAL LENS

A well-designed MCP server can distribute a specialized business capability.

REPOSITORY

`examples/02_tools/mcp_tool_contract.json`

MULTI-AGENT SYSTEMS

Add agents only when specialization or parallel work beats coordination cost.

- Give each agent a distinct role and boundary

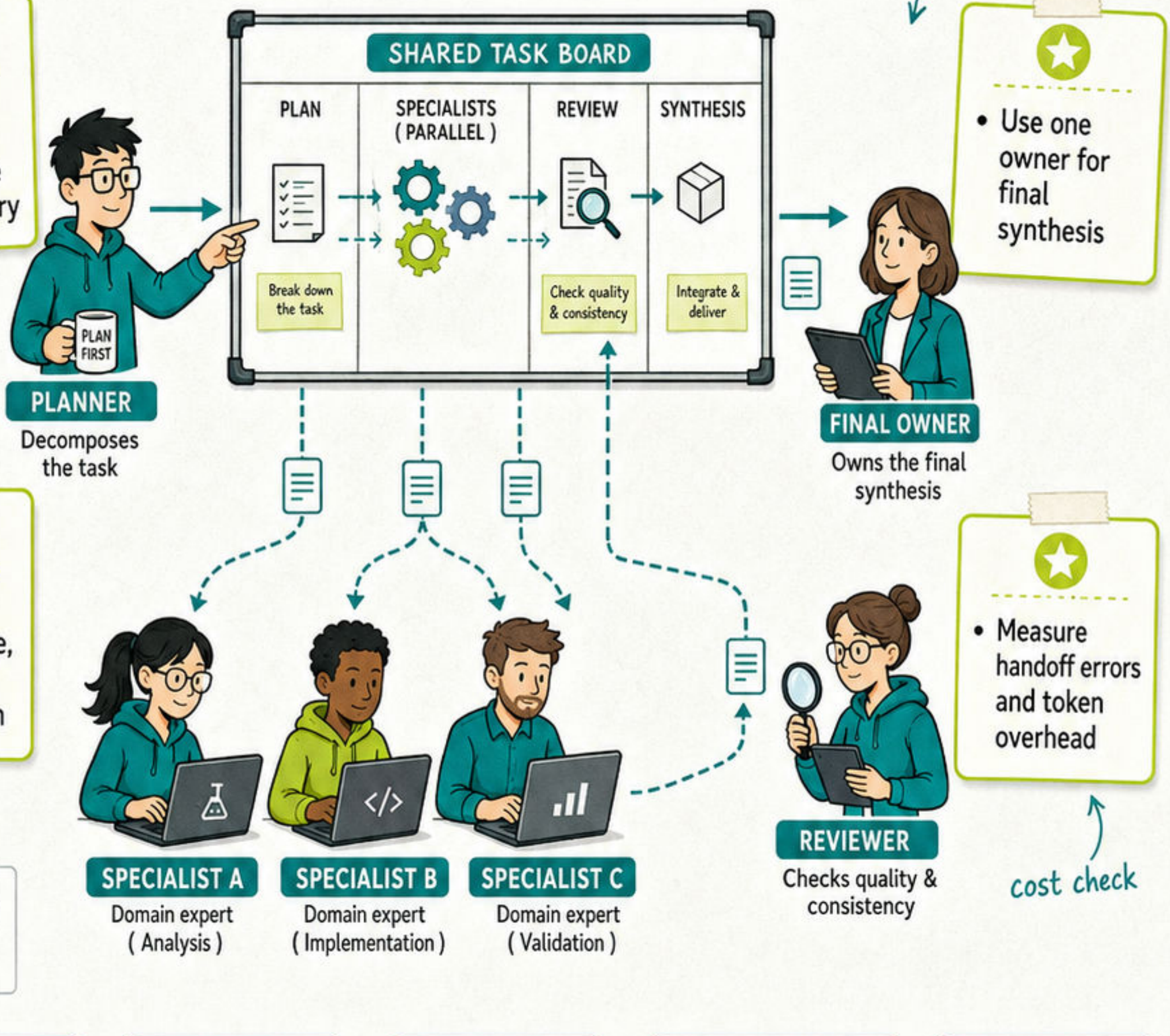
- Share explicit state, not vague conversation

golden rule

- Use one owner for final synthesis

- Measure handoff errors and token overhead

cost check

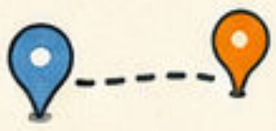


CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`planner → specialists → reviewer → final owner`	Complex decomposable task	Integrated result + trace	Specialist workflows can support high-value research and operations.	`docs/architecture.md`

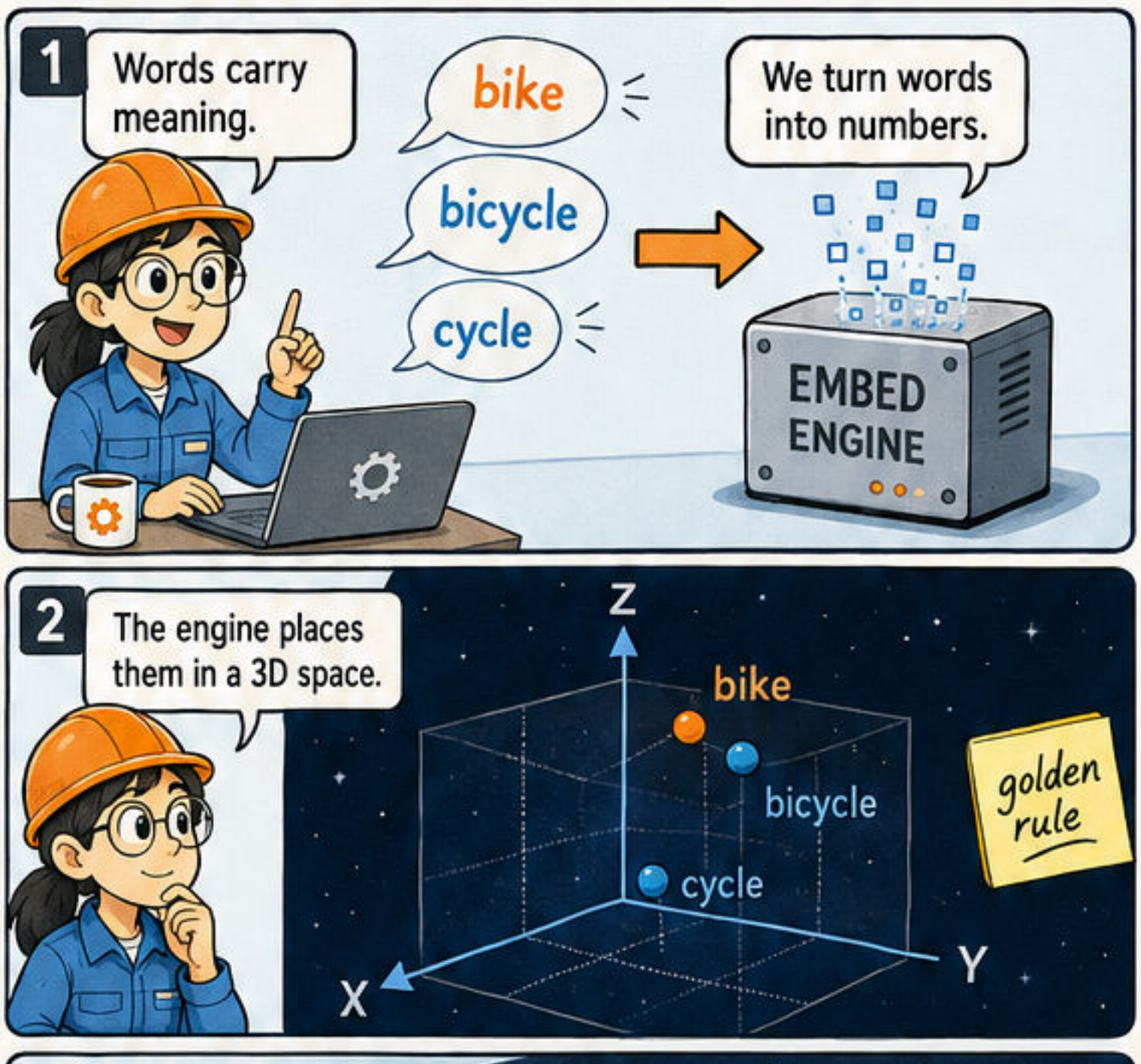
EMBEDDINGS

An embedding maps meaning into a vector that software can compare.

GOLDEN NOTE



Similar meanings tend to have nearby vectors



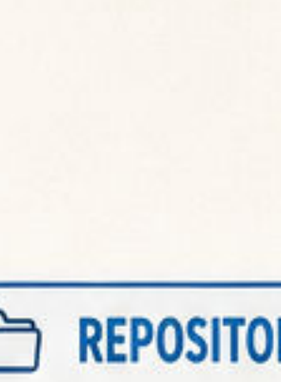
GOLDEN NOTE

Normalize and version embedding pipelines

GOLDEN NOTE

Use the same compatible model for corpus and query

GOLDEN NOTE



Embeddings enable search, clustering and recommendations

CODE

```
vector =
embed("semantic meaning")
```

INPUT

Text or supported content

OUTPUT

[0.21, -0.47, 0.83, ...]

Numeric vector

COMMERCIAL LENS

Semantic infrastructure powers reusable knowledge products.

cost check

REPOSITORY

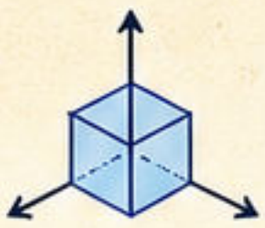
src/ai_api_playbook/rag.py

VECTOR SPACE INTUITION

Distance becomes a practical proxy for semantic relatedness.

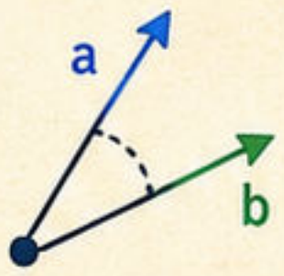
1

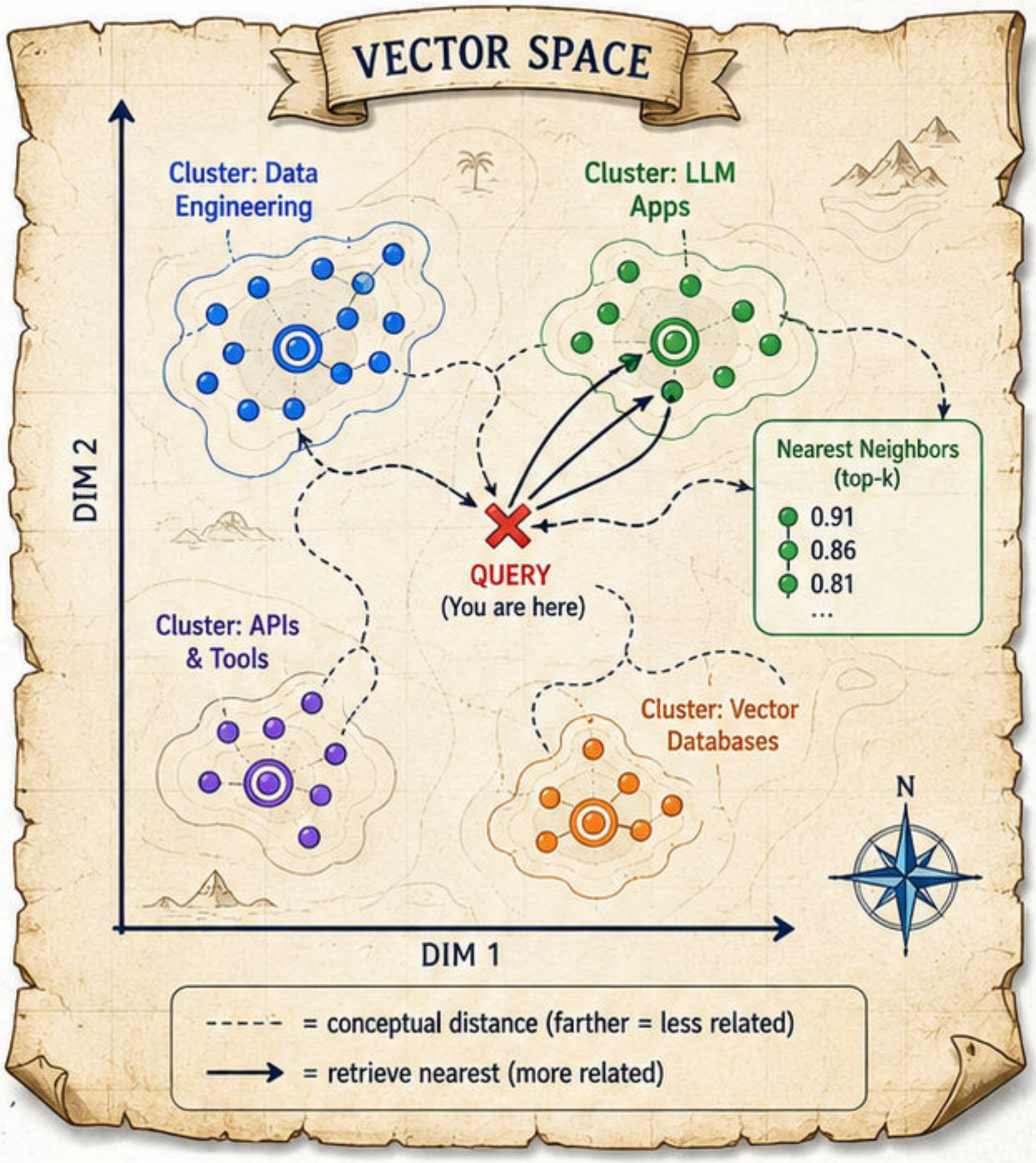
Dimensions are learned features, not human labels



2

Cosine compares direction





3

Distance alone does not prove relevance

4

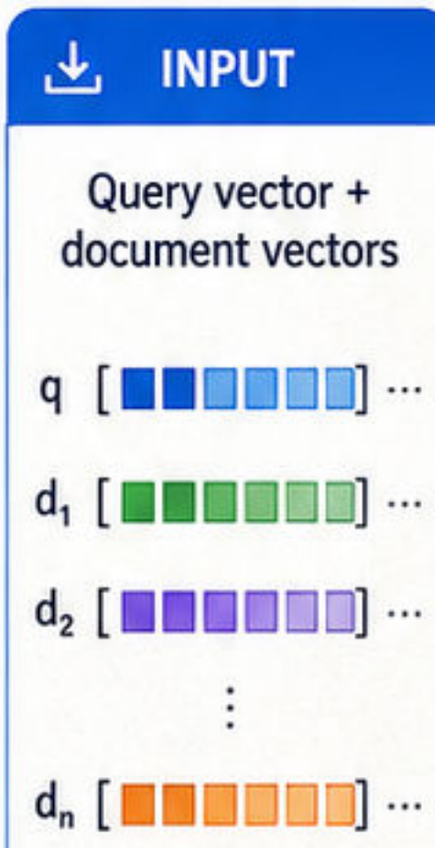
Evaluate retrieval on real user questions

CODE

```
cosine =
dot(a,b)/
(norm(a)*norm(b))
```

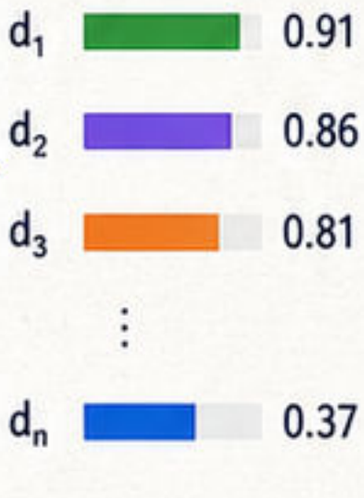
INPUT

Query vector + document vectors

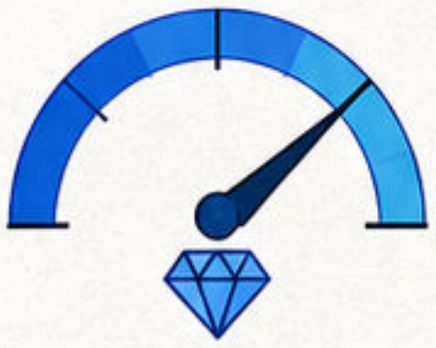


OUTPUT

Similarity scores



COMMERCIAL LENS



Good retrieval increases answer quality without premium generation every time.

cost check ↗

REPOSITORY

src/ai_api_playbook/rag.py

COSINE SIMILARITY

Compare direction when magnitude should not dominate meaning.

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|}$$

A = $[a_1, a_2, \ldots, a_n]$
- direction
- magnitude ignored

B = $[b_1, b_2, \ldots, b_n]$
- direction
- magnitude ignored

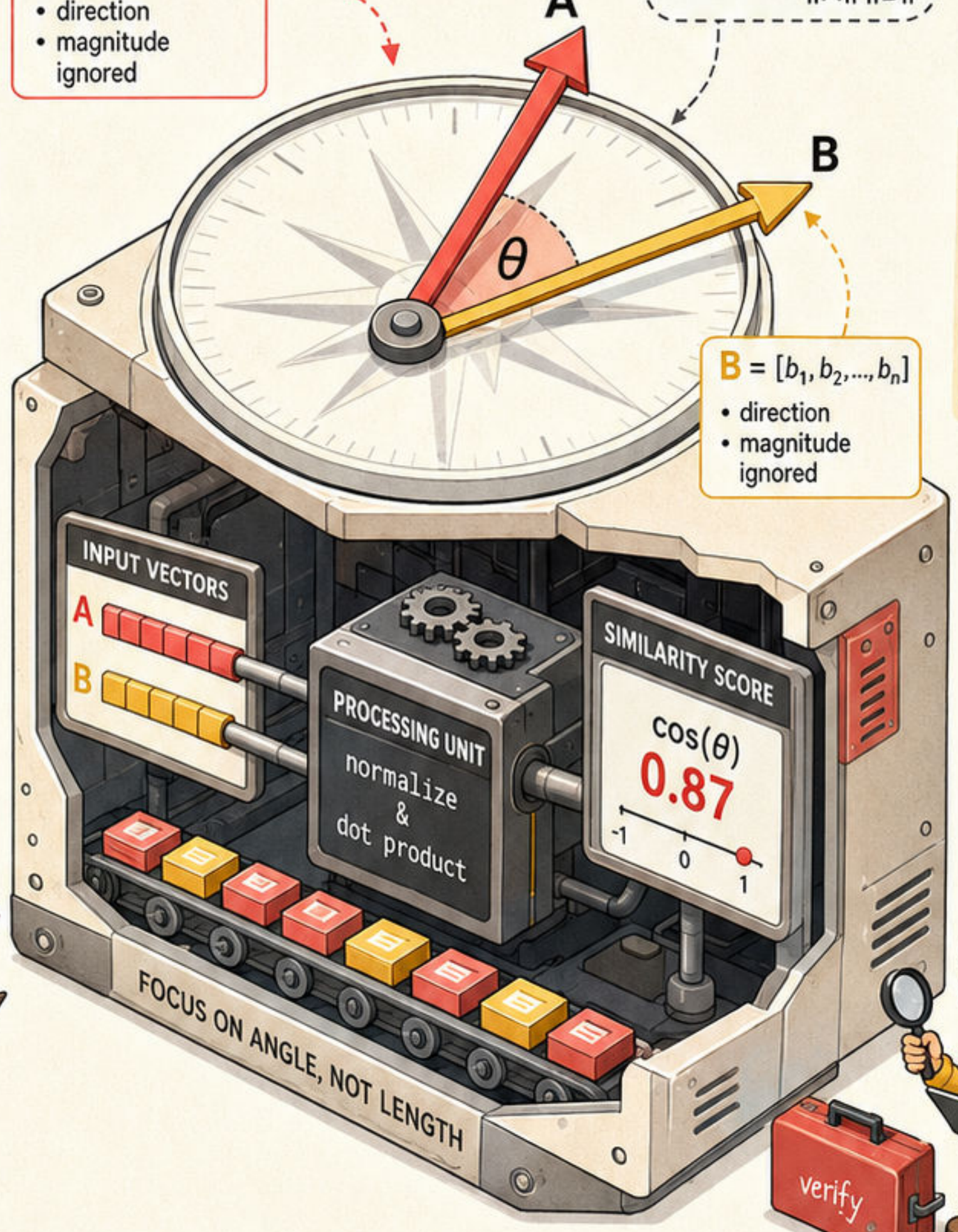
1 A score near one means similar direction

golden rule

2 Zero vectors require explicit handling

3 Thresholds are task-specific

4 Ranking quality matters more than an arbitrary universal cutoff



CODE

```
cosine_similarity([1,0],[1,0])
== 1.0
```

INPUT

[1 , 0]
[1 , 0]

Two equal-length vectors

OUTPUT

1.00

Similarity score

COMMERCIAL LENS

Transparent similarity math builds technical credibility.

REPOSITORY

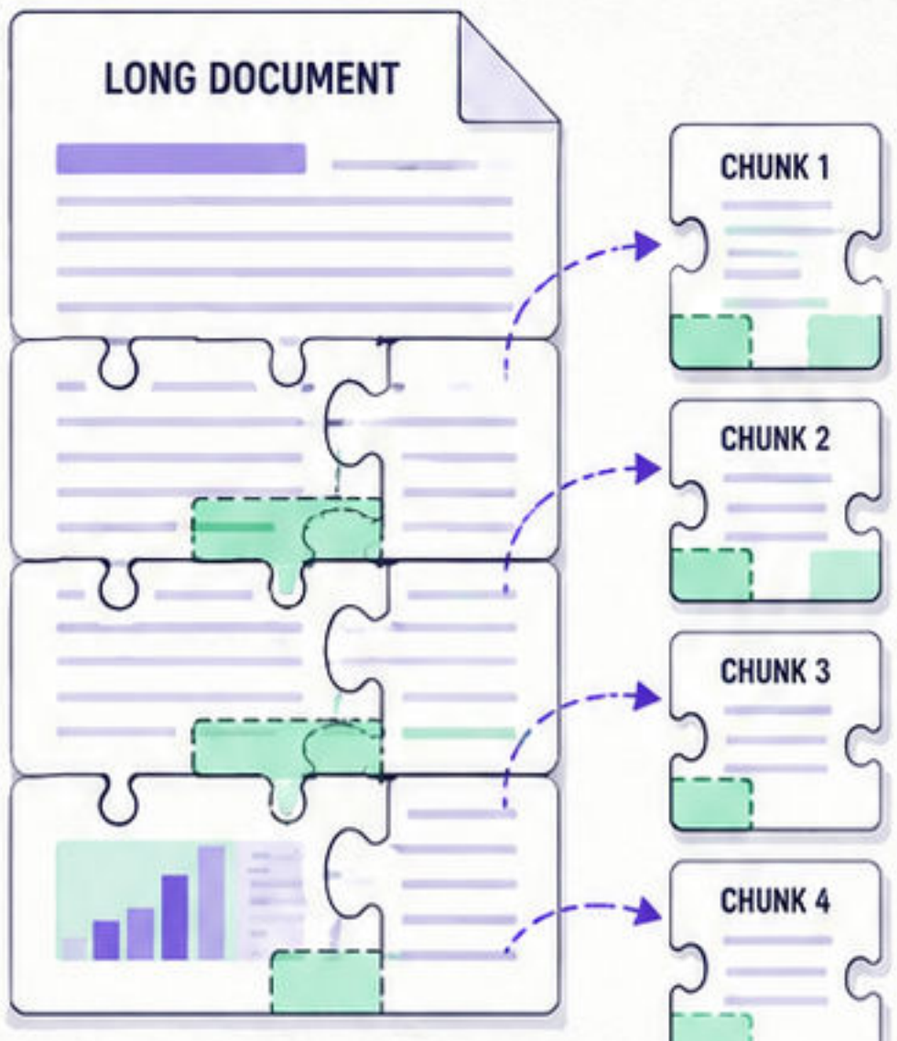
tests/test_core.py

CHUNKING

The chunk is the unit of evidence your retriever can find and your reader can verify.

golden rule

TOO BIG (NO OVERLAP) VS. RIGHT-SIZED (WITH OVERLAP)



Misses cross-boundary meaning and makes verification harder.

Preserves cross-boundary meaning and improves verifiability.

1. Split along semantic boundaries where possible
2. Use overlap to preserve cross-boundary meaning
3. Keep page and heading metadata
4. Test chunk size against retrieval cases

CODE

```
chunks =
 chunk_text(text,
 target_chars=800,
 overlap_chars=120)
```

INPUT

Long document

OUTPUT

Traceable evidence chunks

COMMERCIAL LENS

Better chunking reduces costly re-generation and manual search.

REPOSITORY

`src/ai_api_playbook/rag.py`

METADATA FILTERS

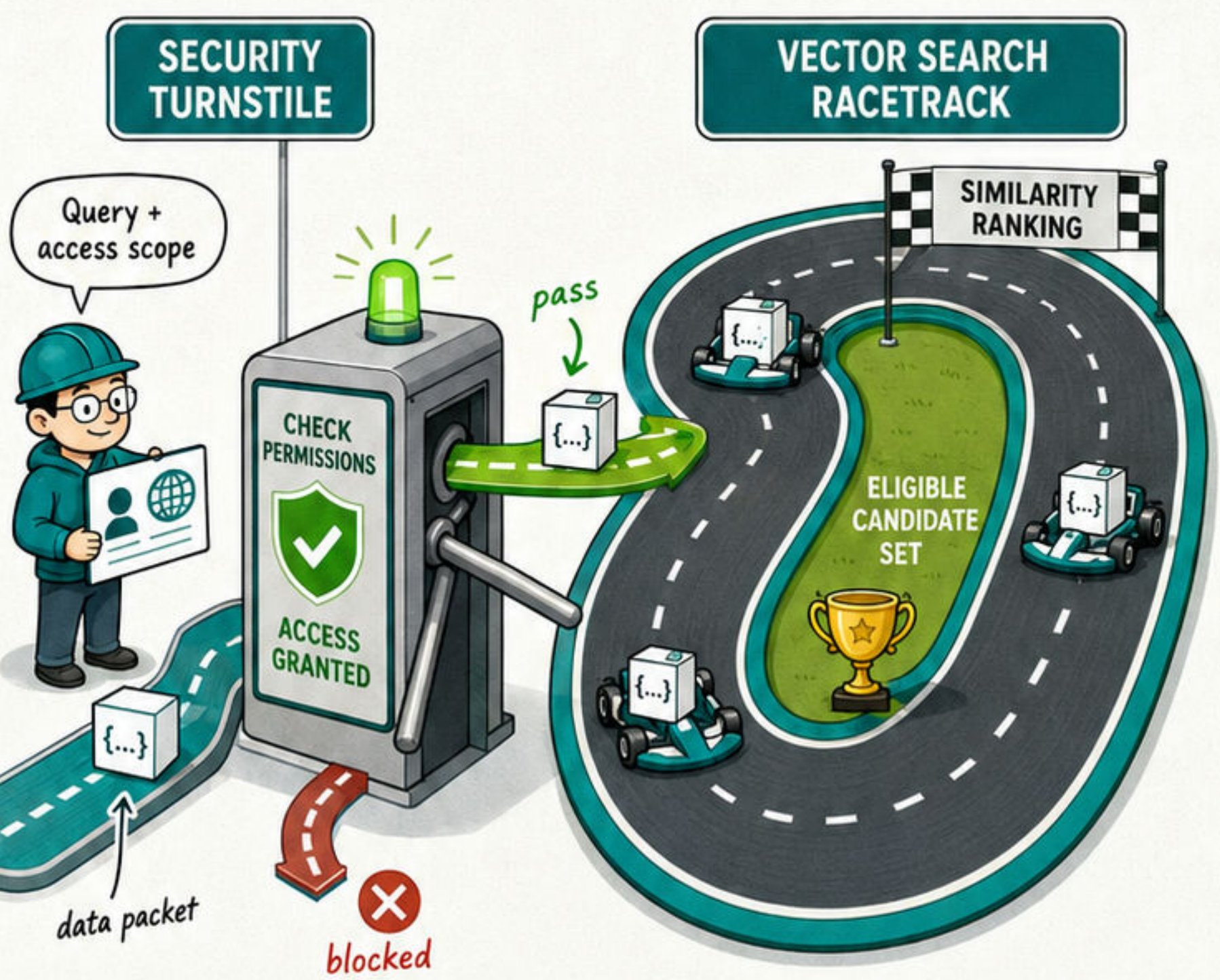
Filter permissions and scope before similarity ranking.

01 Store tenant, language, date, type and access metadata

02 Apply authorization before retrieval

03 Use filters to reduce irrelevant candidates

04 Do not trust model-generated metadata without validation



INPUT	PROCESSING	OUTPUT
Query + access scope	Authorization + Filters	Eligible candidate set

golden rule

CODE

```
filter={"tenant_id":{"$eq": tenant}}
```

INPUT

Query + access scope

OUTPUT

Eligible candidate set

COMMERCIAL LENS

Governed retrieval is essential for enterprise subscriptions.

REPOSITORY

examples/07_search/pinecone_vector_store.py

VECTOR DATABASES

A vector database stores embeddings plus metadata for fast similarity search.

1

Choose managed or self-hosted by scale and governance

2

Plan index dimension and metric

3

Support namespaces or tenant isolation

4

Track deletion and re-embedding workflows

CODE

```
index.upsert(vectors);
index.query(vector, top_k=5)
```

INPUT

Vectors + IDs + metadata

OUTPUT

Ranked matching records

COMMERCIAL LENS

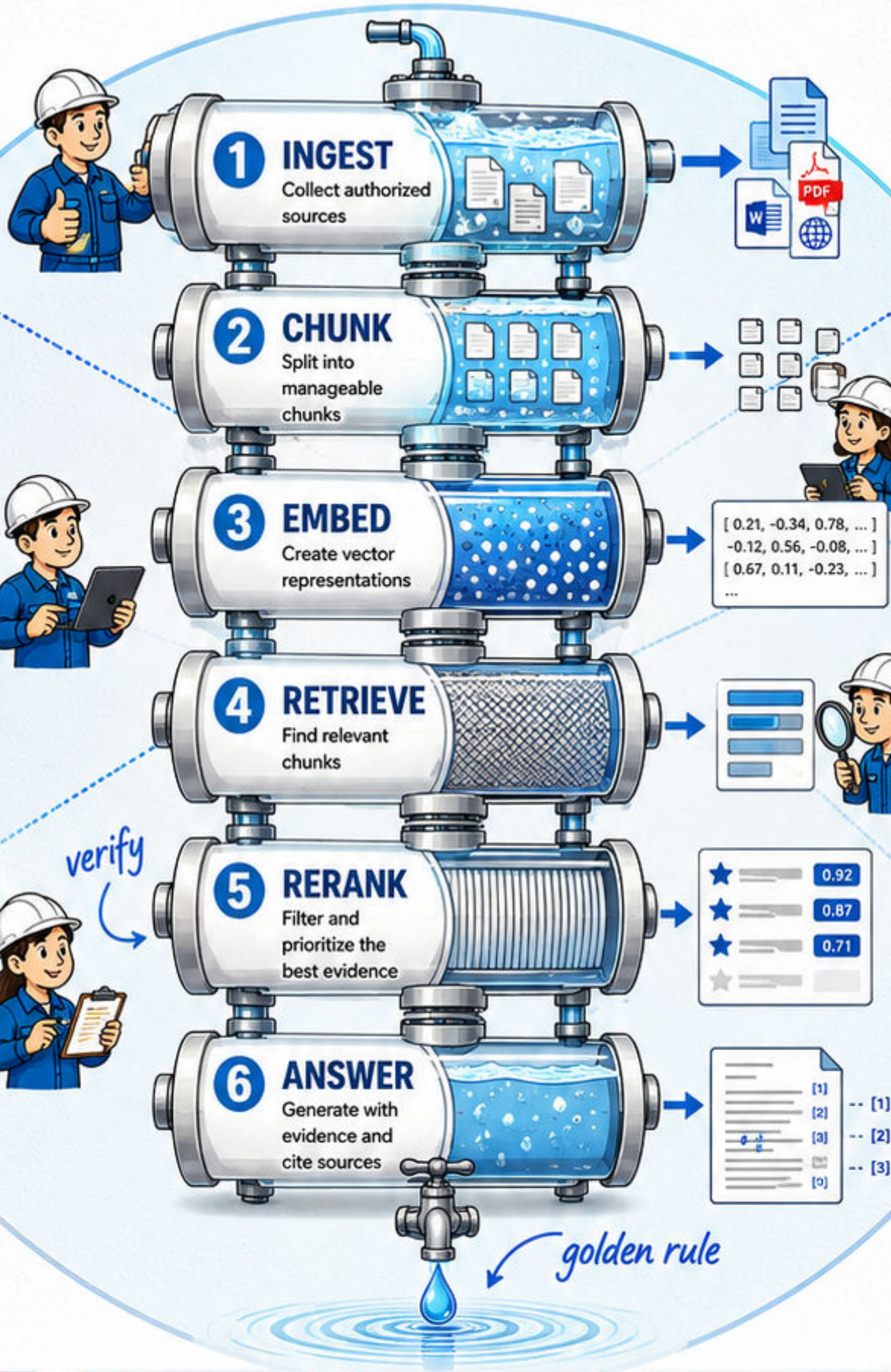
Managed retrieval can anchor recurring knowledge services.

REPOSITORY

examples/07_search/pinecone_vector_store.py

THE RAG PIPELINE

Retrieve evidence first; generate only from the evidence that survives ranking.



GOLDEN NOTE

- Ingest and nornalize authorized sources

GOLDEN NOTE

- Retrieve, filter and rerank

GOLDEN NOTE

- Chunk, embed and index with provenance

GOLDEN NOTE

- Generate with evidence and return citations

CODE

```
ingest → chunk →
embed → retrieve →
rerank → answer
```

INPUT

Question + governed corpus

OUTPUT

Grounded answer + sources

COMMERCIAL LENS

RAG converts private knowledge into a controlled service.

REPOSITORY

`examples/03_rag/local_rag.py`

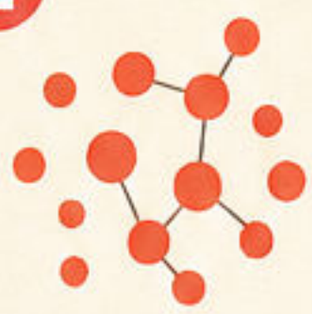
HYBRID SEARCH

Combine semantic understanding with exact lexical matching.

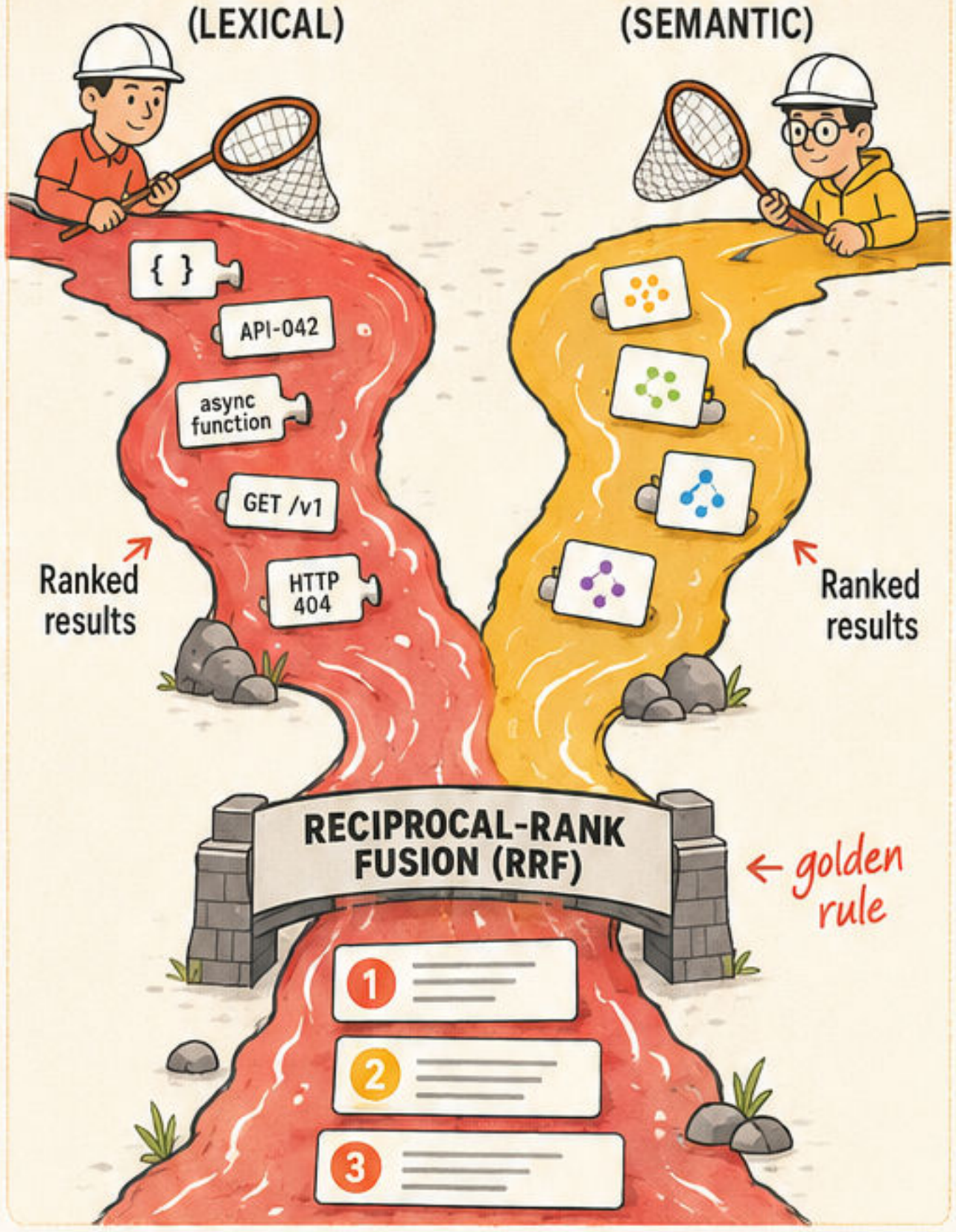
1

Keywords catch names, codes and exact phrases

2



Vectors catch paraphrases and concepts



3

Fuse rankings rather than raw incompatible scores

4

Evaluate by query category

CODE

```python
from typing import List

def rrf(ranks: List[List[int]],
        k: int = 60) -> List[float]:
  scores = {}
  for ranking in ranks:
    for rank, doc_id, in enumerate(
      ranking, start=1):
      scores[doc_id] = scores.get(
        doc_id, 0.0) + 1.0 / (k + rank)
  return sorted(scores.items(),
      key=lambda x: x[1],
      reverse=True)
```

verify

INPUT

Keyword ranking + vector ranking

OUTPUT

Fused candidates

COMMERCIAL LENS

Hybrid search improves reliability in technical and catalog domains.

cost check

REPOSITORY

src/ai_api_playbook/rag.py

RERANKING

Spend a stronger relevance model only on the small candidate set that matters.

1. RETRIEVE BROADLY

Fast, inexpensive retrieval casts a wide net.

2. RERANK WITH A STRONGER MODEL

Judge scores each candidate for relevance to the query and reorders them.

3. PASS BEST EVIDENCE

Only the highest-scoring candidates move forward to the generator.

1. Retrieve broadly and cheaply
2. Rerank top candidates with query-aware scoring
3. Pass only the best evidence to generation
4. Measure precision and end-to-end answer quality

CODE

```
rerank(query,
candidates[:50])[:5]
```

INPUT

Query + candidate passages

OUTPUT

Improved final evidence order

COMMERCIAL LENS

Reranking can lift quality without paying for a larger generator.

cost check

REPOSITORY

`examples/07_search/cohere_rerank.py`

RAG EVALUATION

A fluent answer can still be unsupported, incomplete or badly cited.

1

Measure retrieval recall separately

2

Score groundedness against provided evidence

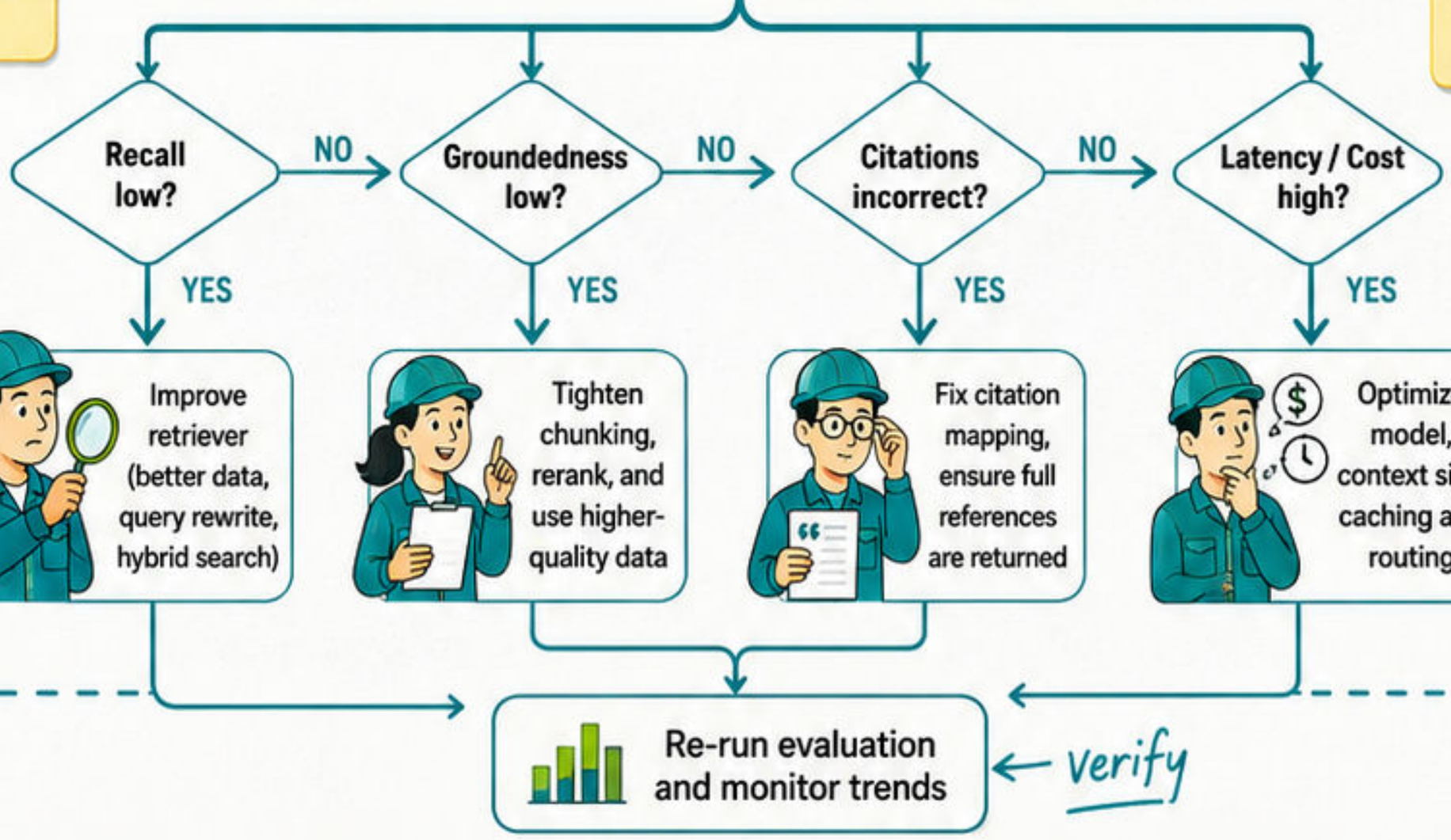
FOUR-GAUGE QUALITY DASHBOARD

3

Check citation correctness and completeness

4

Track latency, cost and abstention behavior



CODE

```
# quality score (0-100)
quality = (.35*grounded +
           .30*complete +
           .20*citation +
           .10*recall +
           .05*cost_penalty)
return quality
```

golden rule

INPUT

Test questions + expected evidence

OUTPUT

Quality, latency and cost report

COMMERCIAL LENS

Evaluation is the difference between a demo and a dependable product.

REPOSITORY

src/ai_api_playbook/evaluation.py

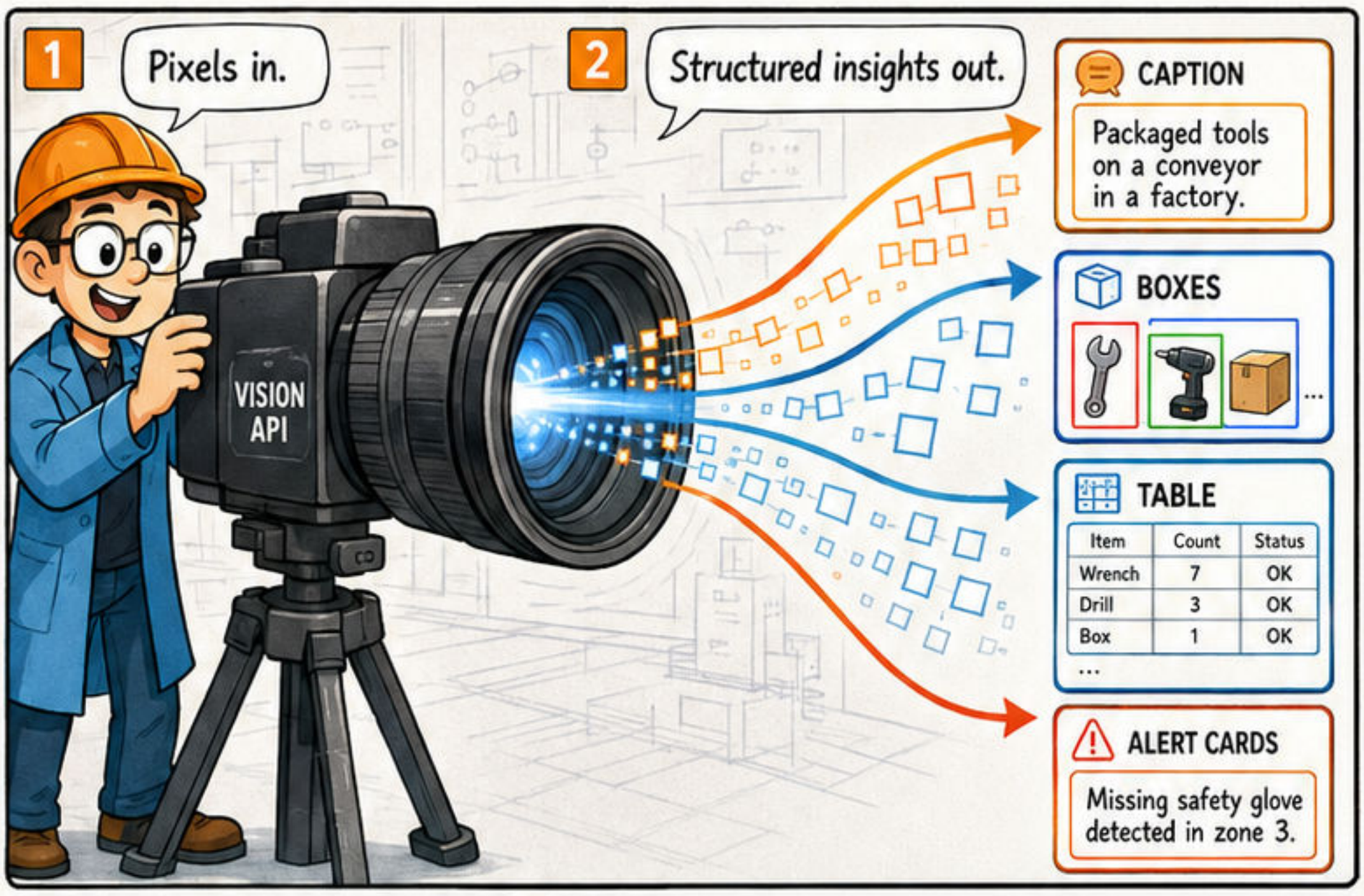
cost check

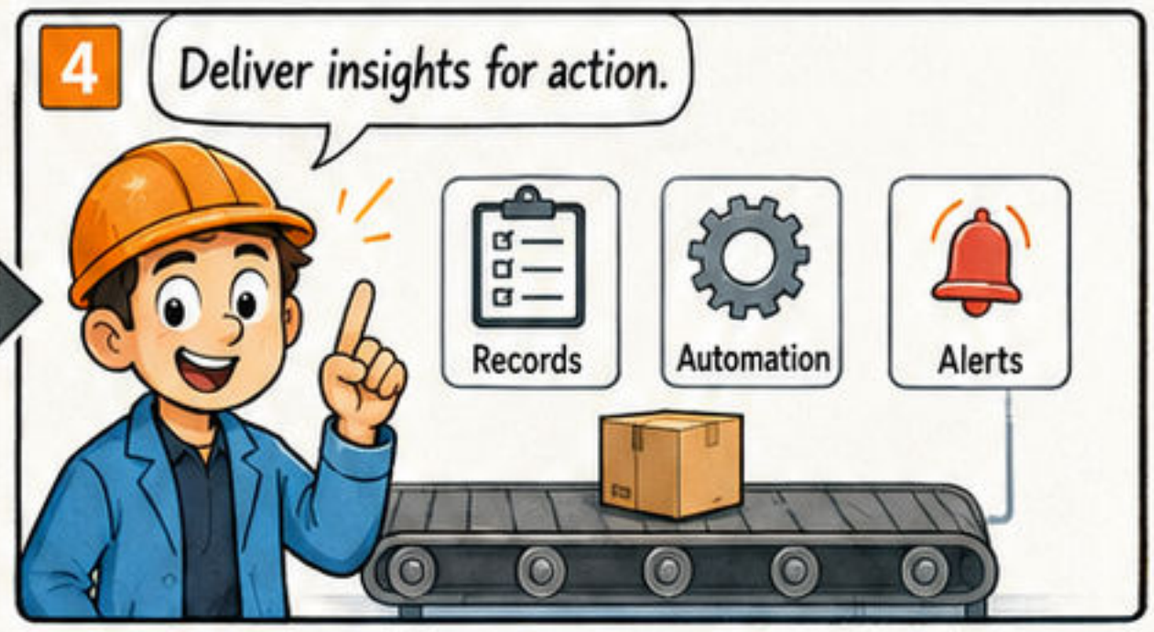
COMPUTER VISION APIS

Vision models translate pixels into descriptions, fields, decisions and actions.

GOLDEN NOTES

- Choose resolution for the task
- Ask precise visual questions
- Preserve original asset and provenance
- Do not infer hidden traits from appearance





CODE

```
1
2 analyze(image,
      task="inventory_count")
3
4
5
6
```

golden rule

INPUT

Authorized image + task

OUTPUT

Description, objects or structured fields

COMMERCIAL LENS

Vertical vision products can automate expensive inspection workflows.

REPOSITORY

docs/
provider-matrix.md

IMAGE UNDERSTANDING

Point the model toward observable evidence, not speculation.

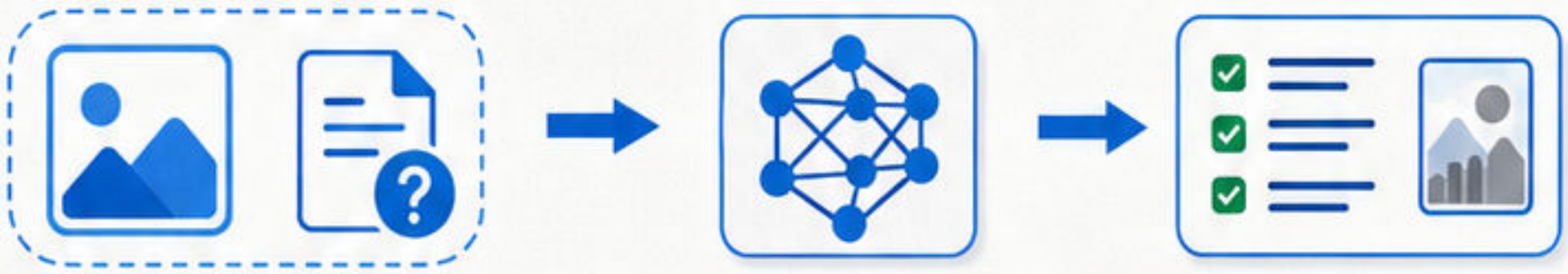
golden rule

Separate observation from inference

Request coordinates or evidence descriptions

Use multiple views for physical tasks

Human-review high-impact conclusions



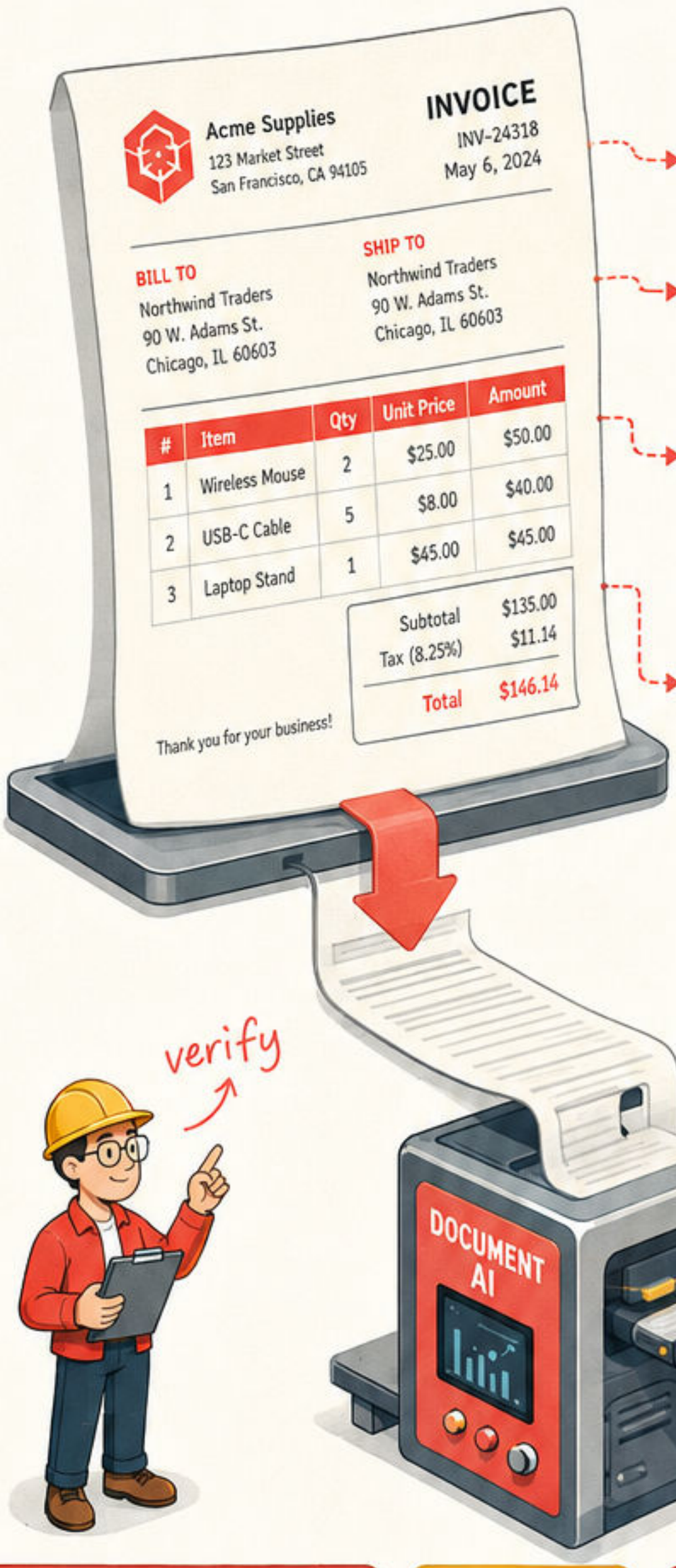
verify

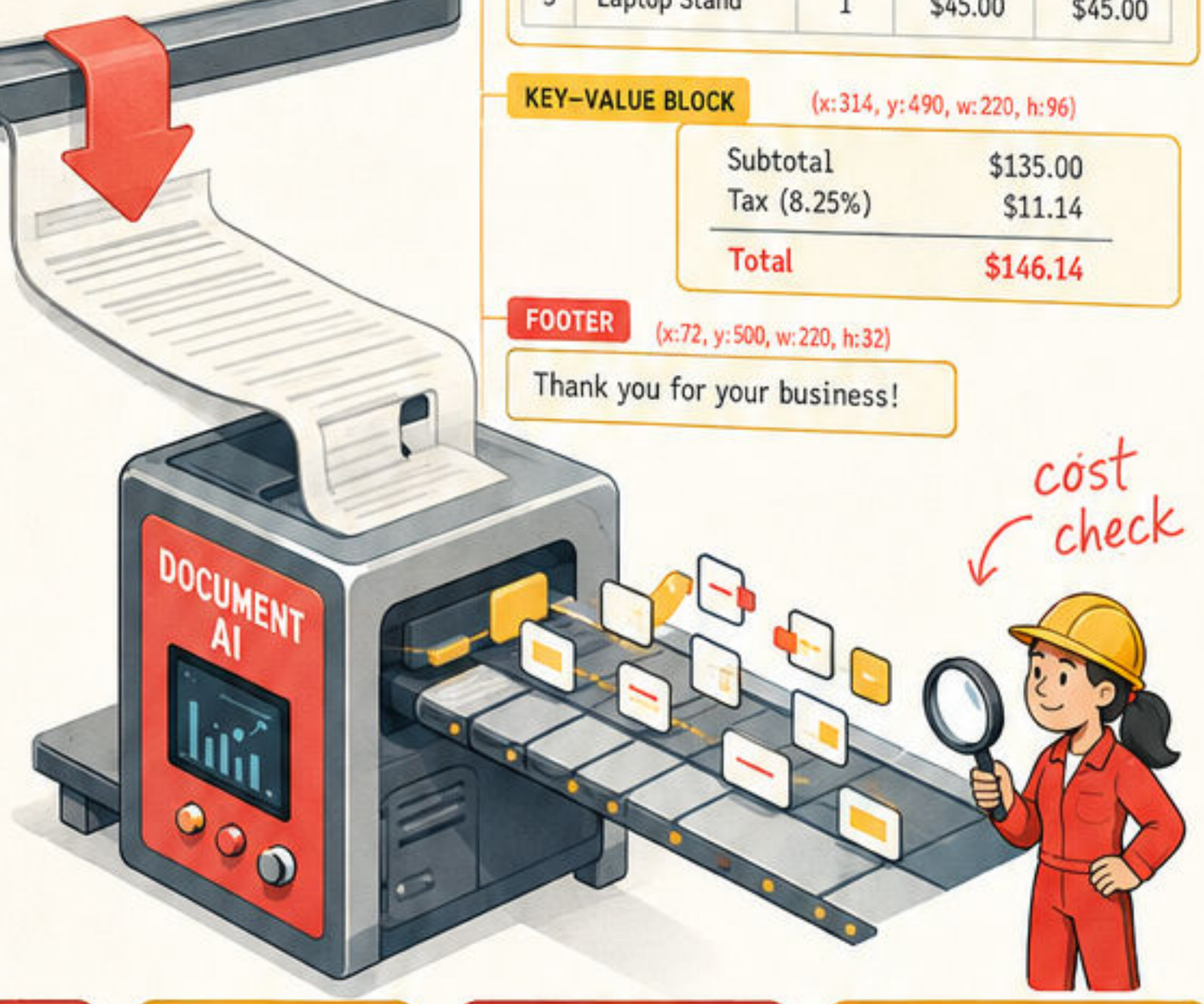
CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`{ "observations": [], "uncertainties": [], "confidence": 0.0 }`	Image + question + schema	Auditable visual observations	Evidence-first vision increases customer trust.	docs/architecture.md

cost check

OCR & DOCUMENT AI

OCR should recover structure, not just a stream of characters.





HEADING (x:72, y:54, w:530, h:48)

INVOICE

KEY–VALUE BLOCK (x:72, y:108, w:240, h:68)

Invoice No: INV-24318
Date: May 6, 2024

BLOCK (x:72, y:190, w:220, h:86)

BILL TO
Northwind Traders
90 W. Adams St.
Chicago, IL 60603

BLOCK (x:314, y:190, w:220, h:86)

SHIP TO
Northwind Traders
90 W. Adams St.
Chicago, IL 60603

TABLE (x:72, y:300, w:462, h:180)

#	Item	Qty	Unit Price	Amount
1	Wireless Mouse	2	$25.00	$50.00
2	USB-C Cable	5	$8.00	$40.00
3	Laptop Stand	1	$45.00	$45.00

KEY–VALUE BLOCK (x:314, y:490, w:220, h:96)

Subtotal $135.00
Tax (8.25%) $11.14
Total $146.14

FOOTER (x:72, y:500, w:220, h:32)

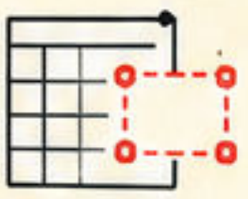
Thank you for your business!

golden rule

1

Preserve pages, reading order and hierarchy

2



Extract tables and coordinates when needed

3

Use confidence to create a review queue

4

Keep source-page provenance for every field

CODE

POST /v1/ocr
→ pages[].markdown
+ blocks[]

```python
import requests

resp = requests.post(
  "https://api.example.com/v1/ocr",
 files={"file": open("doc.pdf","rb")},
 json={"model": "mistral-ocr"}
)

data = resp.json()
print(data["pages"][0]["markdown"])
print(data["blocks"])
```

INPUT

PDF or scan

OUTPUT

Markdown, tables, blocks and coordinates

```
{
  "type": "table",
  "bbox": [72,300,462,180],
  "confidence": 0.98
}
```

COMMERCIAL LENS

Document extraction saves measurable manual re-keying time.

REPOSITORY

examples/04_documents/
mistral_ocr.py

PDF UNDERSTANDING

A PDF can contain text, scans, tables, charts and misleading reading order.

01

Detect native text versus scanned pages

02

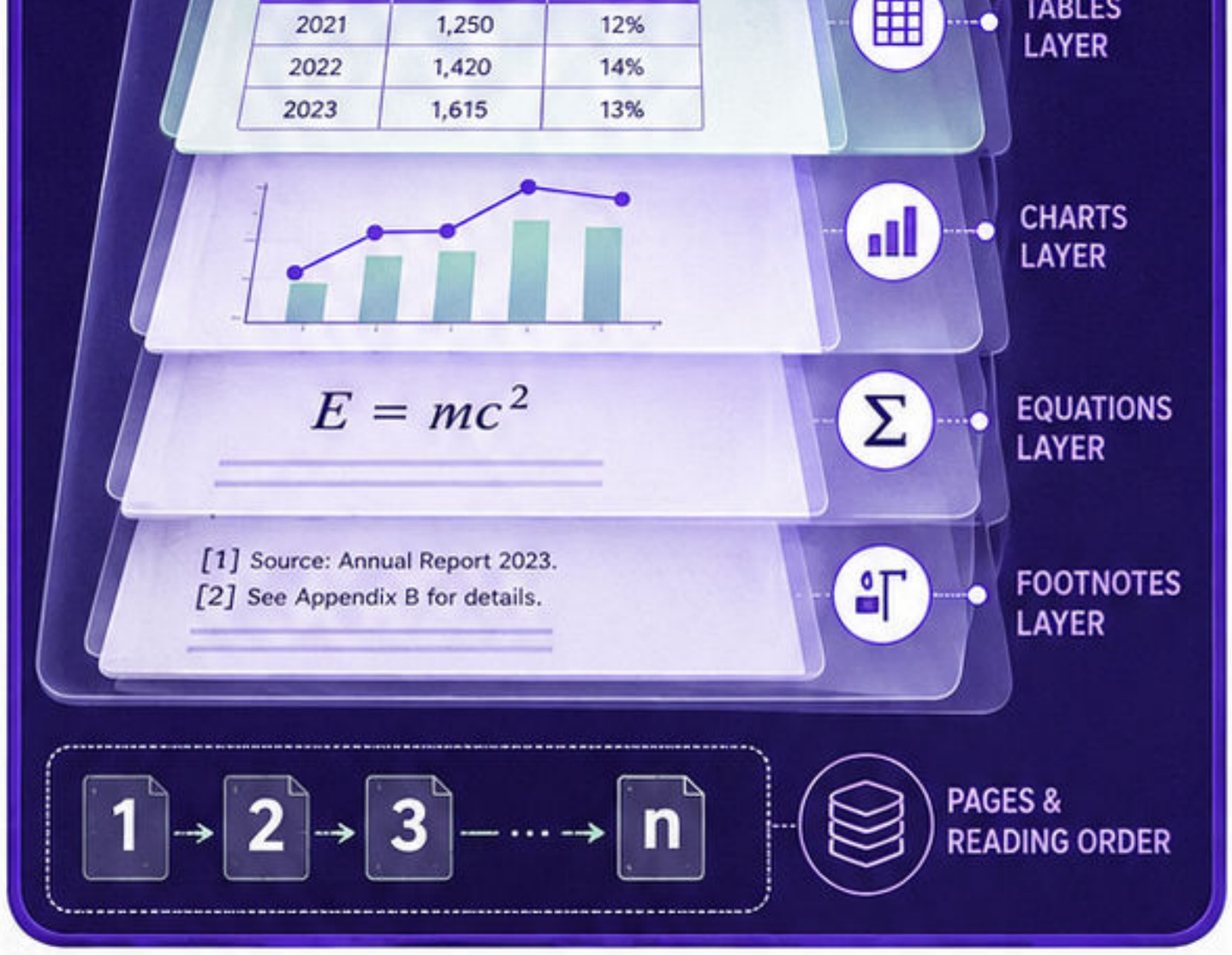
Render pages when layout carries meaning

03

Chunk by heading and page

04

Validate tables, equations and footnotes separately



CODE

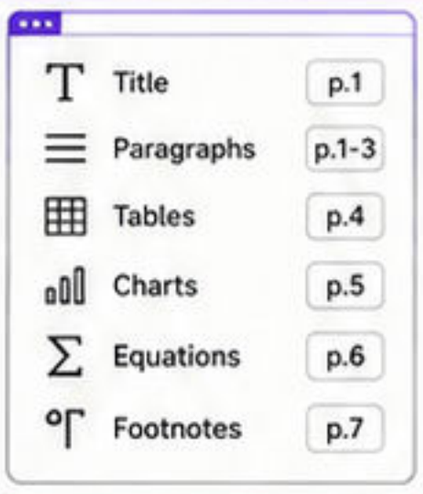
```python
import pdfplumber
import pytesseract

def pipeline(path):
    doc = pdfplumber.open(path)
    for page in doc.pages:
        if page.extract_text():
            text = page.extract_text()
        else:
            image = page.to_image()
            text = pytesseract.image_to_string(
            image.original)
    # structure by heading/page
    data = structure_page(text, page.page_number)
    verify(data)
    return data
```

pdf → inspect → OCR/render → structure → verify

INPUT

Authorized PDF

OUTPUT



Structured document with page references

COMMERCIAL LENS

Reliable PDF pipelines serve publishing, accounting and research.

REPOSITORY

projects/publishing_engine/pipeline.py

STRUCTURED DOCUMENT EXTRACTION

Every extracted value needs a type, evidence location and confidence path.

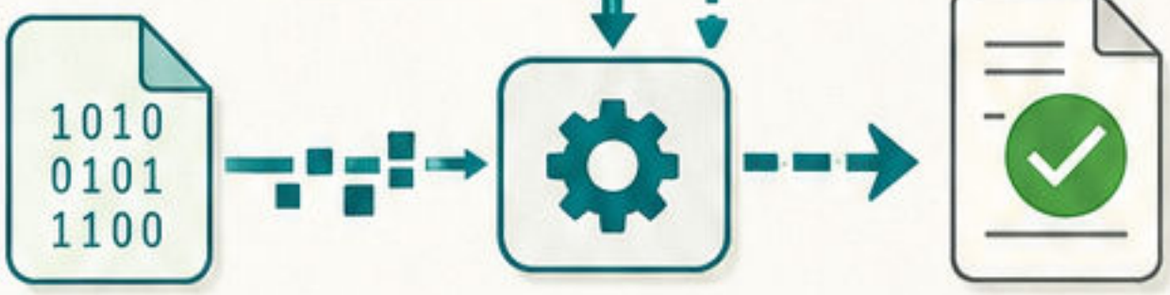
1. Define a strict domain schema

2. Attach page and bounding-box evidence

CUSTOMS DECLARATION FORM

Importer Name	Acme Global Ltd.	VALID P3
Importer ID	IMP-883721	VALID P3
Date of Import	2024-05-18 (YYYY-MM-DD)	VALID P3
Total Invoice Amount	USD 12,450.00	VALID P3
Country of Origin	Germany	VALID P3
HS Code	8471.30??	REVIEW P3
Duty Rate	7.5%?	REVIEW P3
Additional Notes	See attached sheet.	REVIEW P5

Evidence Tag
page: 3
bbox: [120, 245, 420, 268]



3. Normalize dates, currencies and identifiers

Send low-confidence or conflicting fields to review

cost check

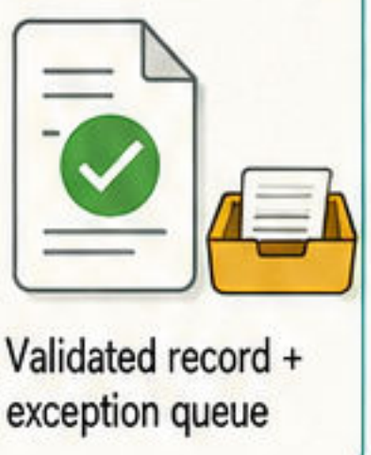
CODE

```
{
  "value":"...",
  "page":3,
  "confidence":0.91
}
```

INPUT

Document pages

OUTPUT

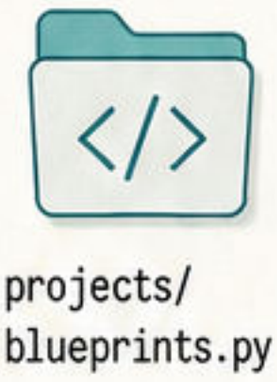


Validated record + exception queue

COMMERCIAL LENS

Charge for successful verified documents, not raw OCR pages alone.

REPOSITORY

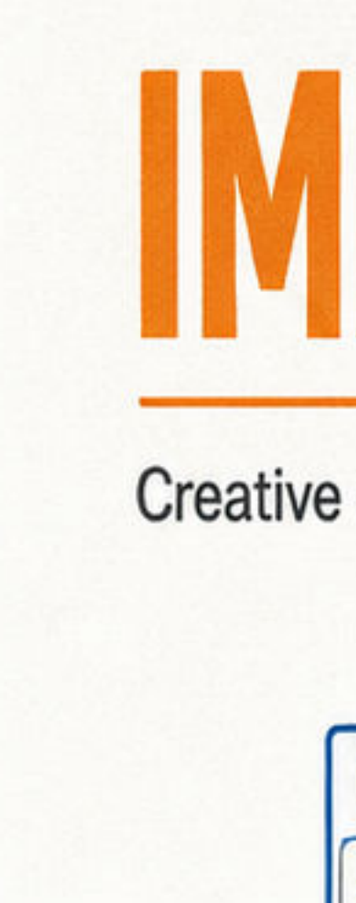


`projects/blueprints.py`

Validated path (accepted) · Review path (exception) · Bounding-box evidence

IMAGE GENERATION APIS

Creative output becomes useful when the brief controls purpose, format and review.



1 PROMPT STORYBOARD

SUBJECT
Solar-powered backpack

COMPOSITION
Three-quarter view on a rocky trail with mountains

MEDIUM
Photorealistic product photo

CONSTRAINTS

- ☑ No visible logos
- ☑ A4 vertical
- ☑ Natural lighting
- ☑ Include shadow

01010110
10110011
11010101
00101110

DATA PACKET

2 ASSET VARIANTS (3)

1 · A4 · Photo · v1 · 2025-05-20 · # id: 8f3a...21

2 · A4 · Photo · v2 · 2025-05-20 · # id: d4b7...90

3 · A4 · Photo · v3 · 2025-05-20 · # id: e9c1...77

Specify subject, composition, medium and constraints

Use references only with proper rights

Generate variants for selection

Inspect text, anatomy, brands and factual claims

REVIEW CHECKLIST

- ☑ Text clarity
- ☑ Anatomy & realism
- ☑ Brand compliance
- ☑ Factual accuracy

verify

cost check

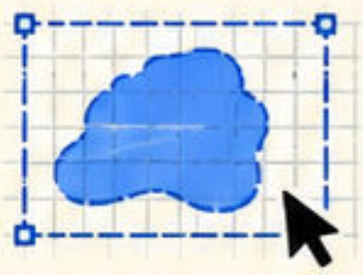
CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`generate(prompt, size="A4", variants=3)`	Creative brief + authorized references	Candidate images + metadata	Productized creative services can sell outcome packages.	examples/05_media/ openai_image_generation.py

IMAGE EDITING

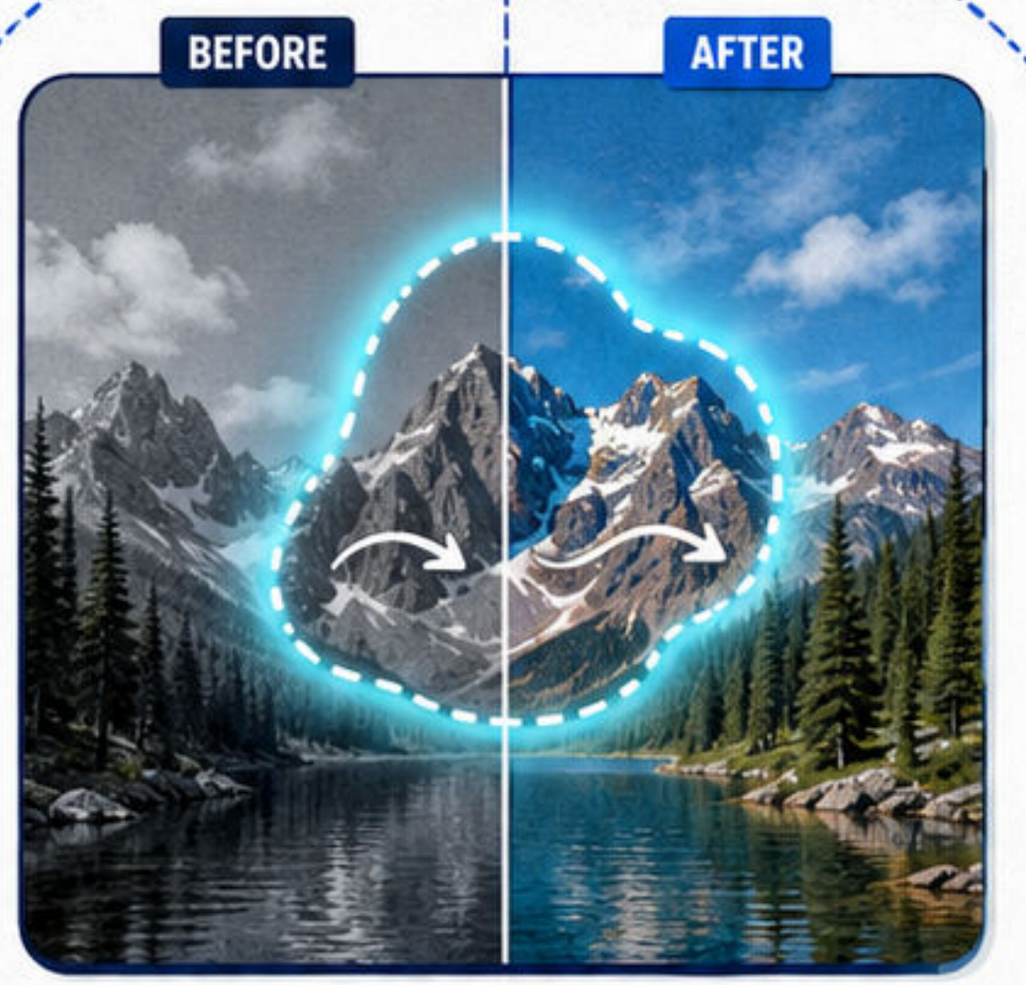
Editing requires both a source image and an explicit change boundary.

1 State what must remain unchanged

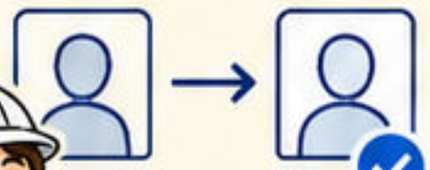
2 Use masks or regional instructions



CONTROLLED CHANGE



3 Preserve aspect ratio and identity only with consent



golden rule

4 Compare output against the source at full resolution

verify

CODE

```
edit(image, mask, instruction)
# image: source image
# mask : change boundary
# instruction:  edit brief

response = client.images.edit
(
   image=image,
   mask= mask,
   instruction=instruction,
)
edited = response.output[0]
```

INPUT

Source + mask + edit brief

OUTPUT

Edited candidate

COMMERCIAL LENS

Controlled editing supports catalogs, marketing and restoration workflows.

cost check

REPOSITORY

examples/05_media/
openai_image_generation.py

VISUAL QUALITY ASSURANCE

Generated does not mean publishable.

Check spelling and typography at 100% zoom

Inspect crops, hands, faces and repeated elements

Verify logos, products and factual labels

Record seed, prompt, model and human approval

CODE

```
generate →
render →
inspect →
revise →
approve
```

INPUT

"Candidate visual"

OUTPUT

"Approved asset + provenance"

COMMERCIAL LENS

QA protects brand reputation and reduces refunds.

cost check

REPOSITORY

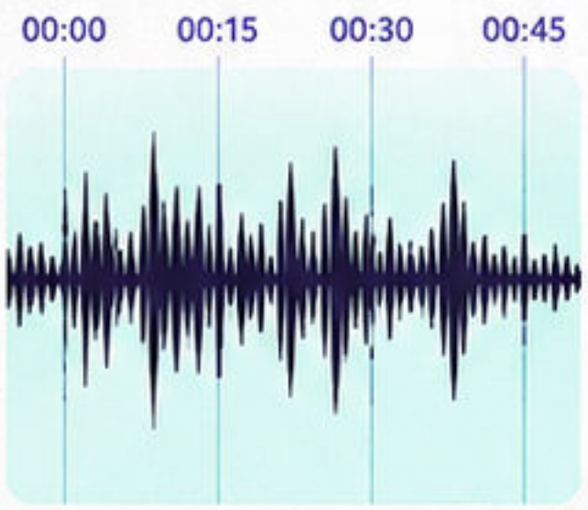
{ } SECURITY.md

SPEECH TO TEXT

Transcription turns audio into searchable, time-aligned data.

1. AUDIO INPUT

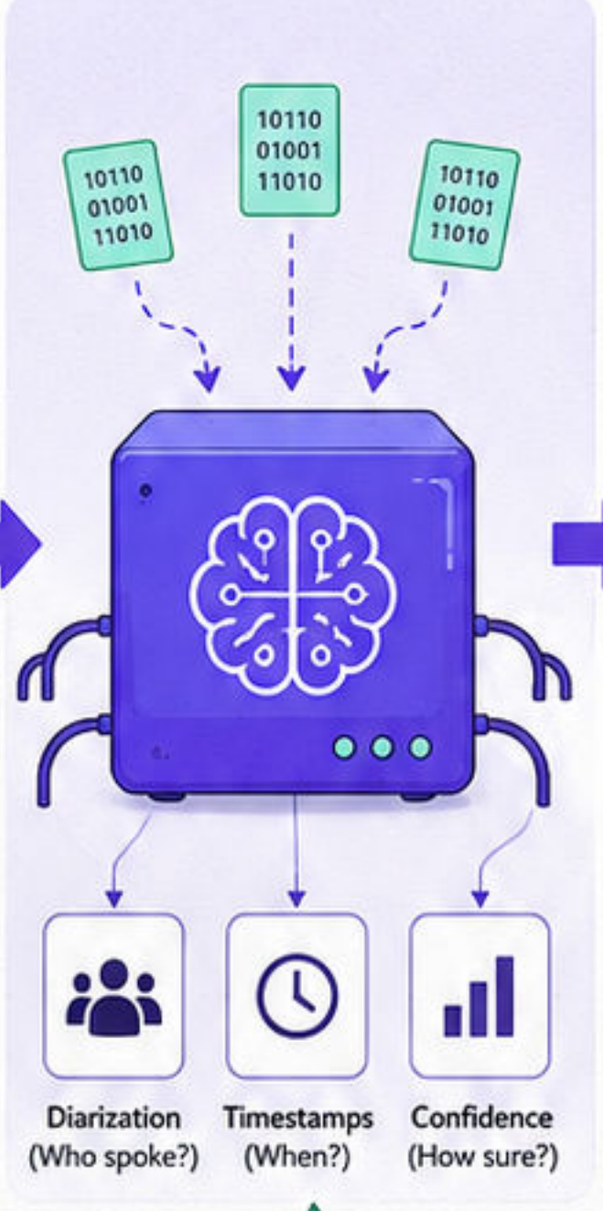
00:00 00:15 00:30 00:45



Live / Streaming

File Upload

2. TRANSCRIPTION ENGINE



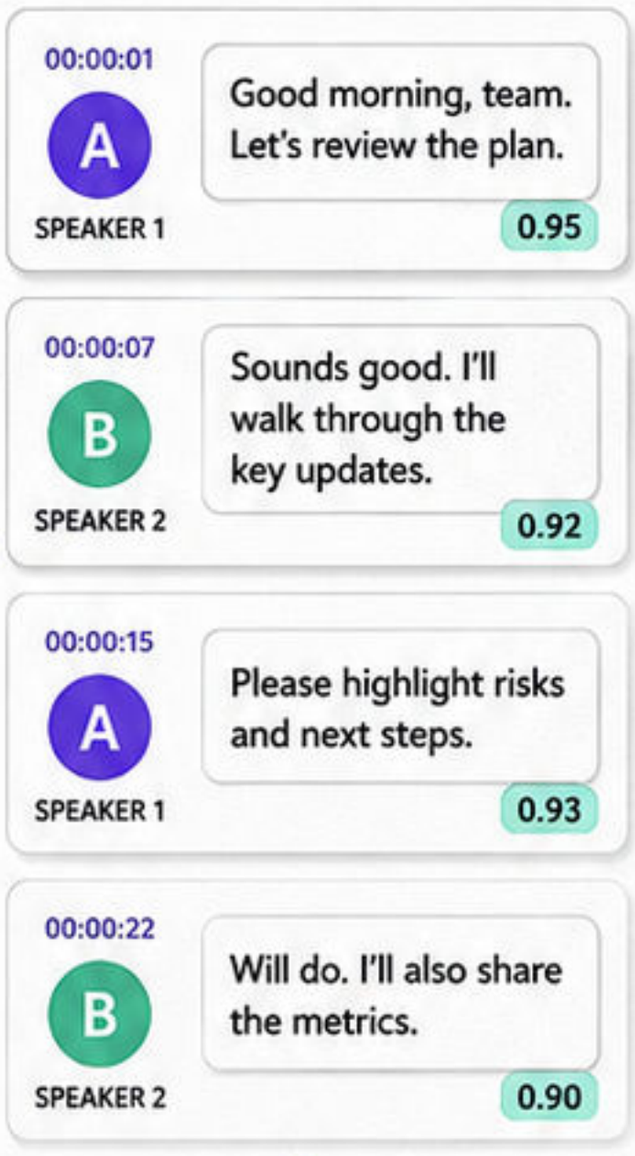
Diarization (Who spoke?)

Timestamps (When?)

Confidence (How sure?)

verify

3. TIME-ALIGNED OUTPUT



1 Choose prerecorded or streaming mode

2 Request timestamps and diarization when needed

3 Handle accents, noise and domain vocabulary

4 Keep confidence and original audio for review

golden rule

CODE

```
POST /v1/listen?diarize=true
```

INPUT

Audio stream or file

OUTPUT

Transcript + speakers + timestamps

COMMERCIAL LENS

Transcription supports meetings, media and support analytics.

REPOSITORY

examples/05_media/deepgram_stt.py

</>

TEXT TO SPEECH

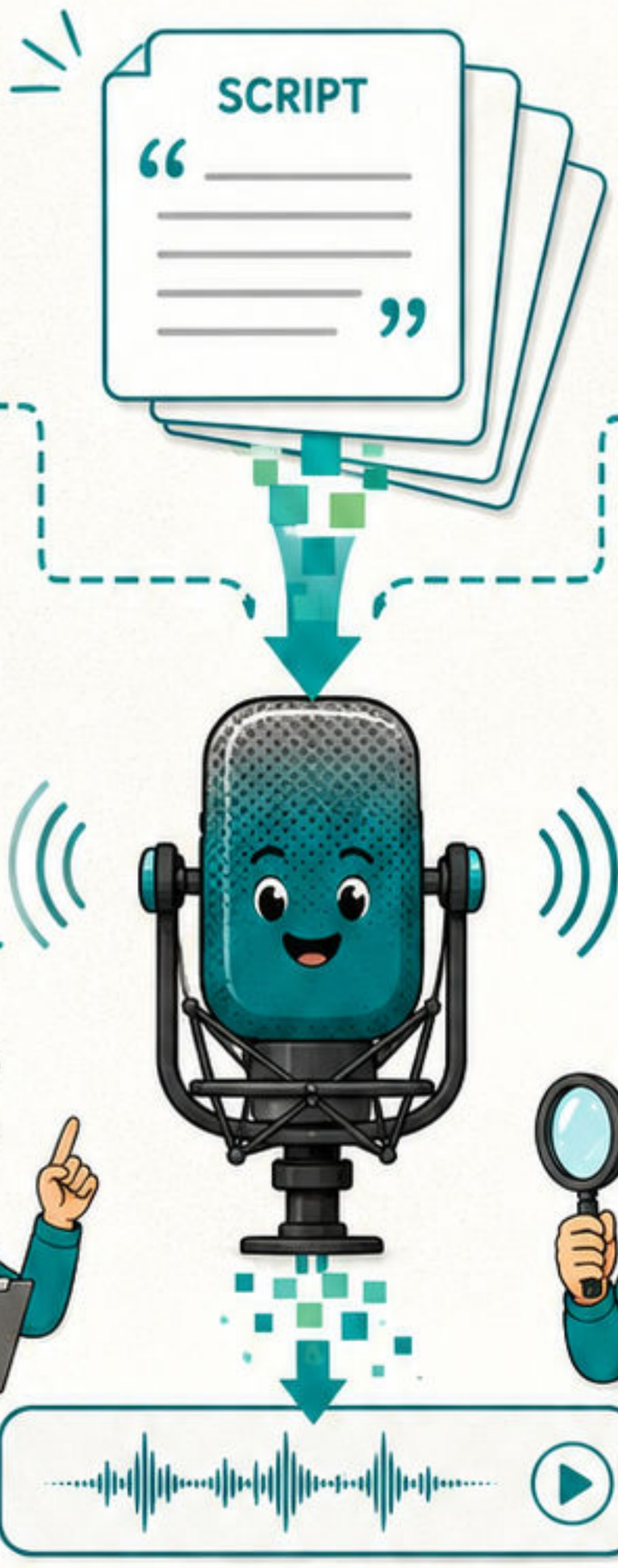
A voice API converts approved text into controlled audio delivery.

1 Choose language, voice and speaking style

2 Stream when text arrives incrementally

3 Normalize numbers, dates and abbreviations

4 Disclose synthetic speech where appropriate



CODE

```
import requests
import json

url = f"https://api.
 elevenlabs.io/v1/
 text-to-speech/{voice_id}"

headers = {
   "xi-api-key":  API_KEY,
   "Content-Type":
    "application/json"
}
payload = {
  "text": TEXT,
  "model_id": "eleven_multilingual_v2"
}
response = requests.post(
   url, headers=headers,
   json=payload, stream=True
)
forchunk in response.iter_content(
  chunk_size=1024):
    if chunk:
       save_chunk(chunk)
```

INPUT

Text + voice settings

OUTPUT

Audio bytes or stream

COMMERCIAL LENS

TTS enables audiobooks, learning and accessible content.

golden rule

REPOSITORY

examples/05_media/
elevenlabs_tts.py

REALTIME VOICE

A natural voice product is a latency system before it is a language demo.

Measure time to first audio

golden rule

Keep session state bounded

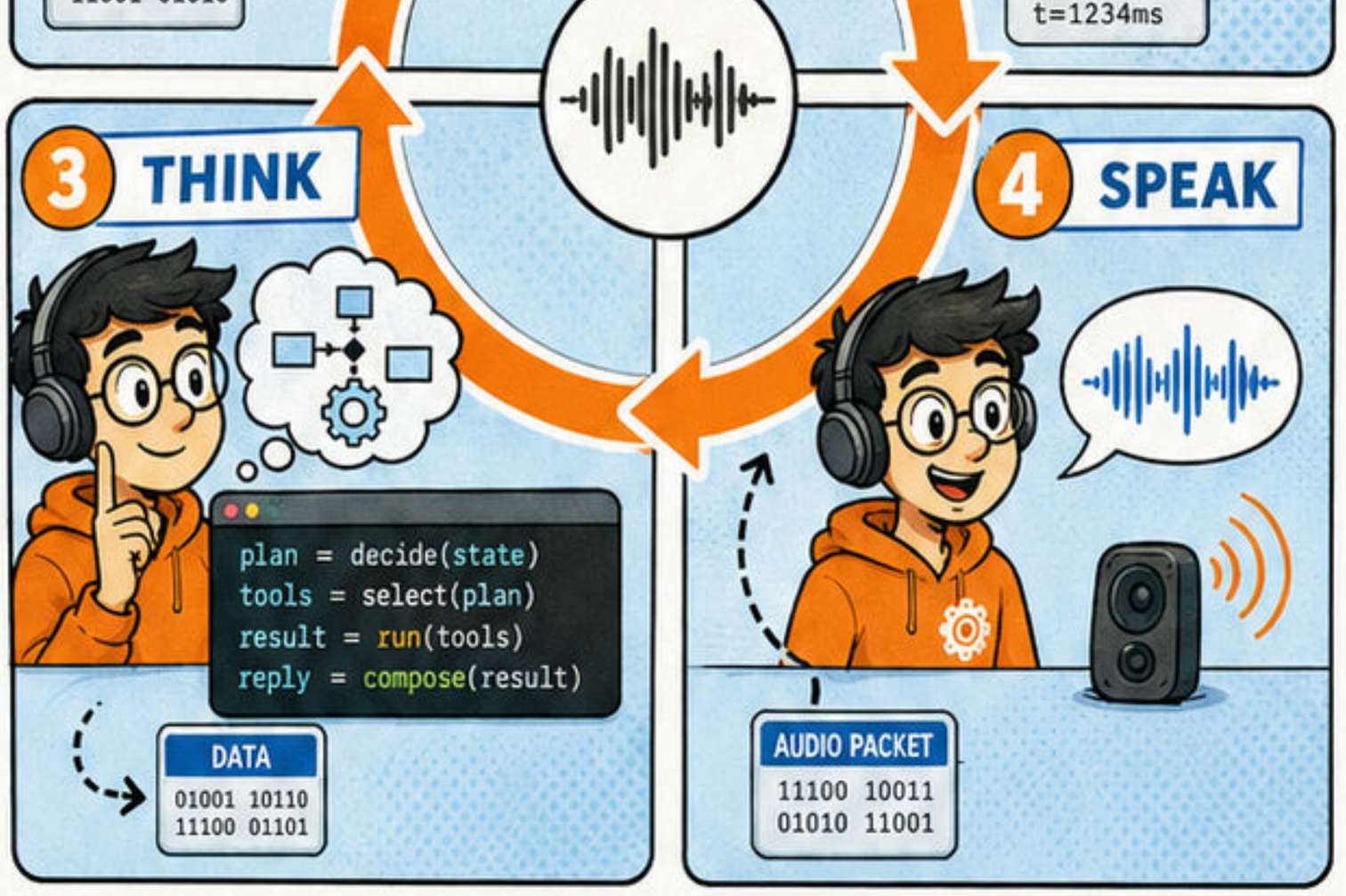
golden rule

Support interruption and barge-in

golden rule

Use tools without blocking the conversation

golden rule



CODE

```
session = client.realtime.connect()
while session.is_open():
    audio_in = session.receive_audio()
    event = session.detect_turn(audio_in)
    if event == "turn_start":
        response = session.generate()
        for chunk in response.audio_out():
            session.send_audio(chunk)
session.close()
```

audio in ↔ realtime session ↔ audio out

INPUT

Live microphone audio

OUTPUT

Transcript, tool events and speech

COMMERCIAL LENS

Realtime voice supports premium assistants and reception services.

REPOSITORY

docs/
provider-matrix.md

VOICE AGENT ARCHITECTURE

Listening, reasoning, tools and speaking must share one controlled session.

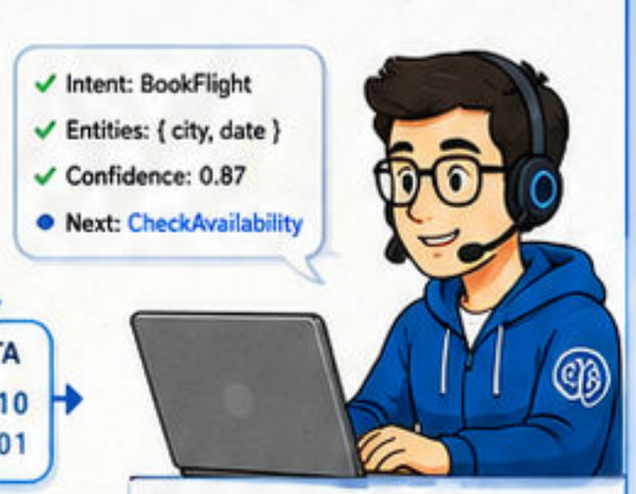
golden rule

- Authenticate the caller and session

1 STT
Speech-to-Text

DATA
10101
01010

2 REASONING AGENT
Understand and Decide



DATA
11010
00101

golden rule

- Separate STT, reasoning, tools and TTS metrics

SHARED SESSION STATE HUB

- Identity
- Context
- History
- Policies
- Evidence

golden rule

- Confirm names, dates and irreversible actions

3 TOOLS
APIs and Systems

DATA
01100
10110

4 TTS
Text-to-Speech

DATA
10011
01001

golden rule

- Escalate uncertainty to a person

INPUT
Caller speech

PROCESSING
STT → agent → tool → TTS

OUTPUT
Answer, booking result and call summary

</> CODE

```
STT → agent →
tool → TTS
```

INPUT
Caller speech

OUTPUT
Answer, booking result and call summary

COMMERCIAL LENS
Voice agents monetize through setup, subscription and minutes.

REPOSITORY
projects/blueprints.py

TURN-TAKING & INTERRUPTION

The best answer fails if the agent speaks over the customer.

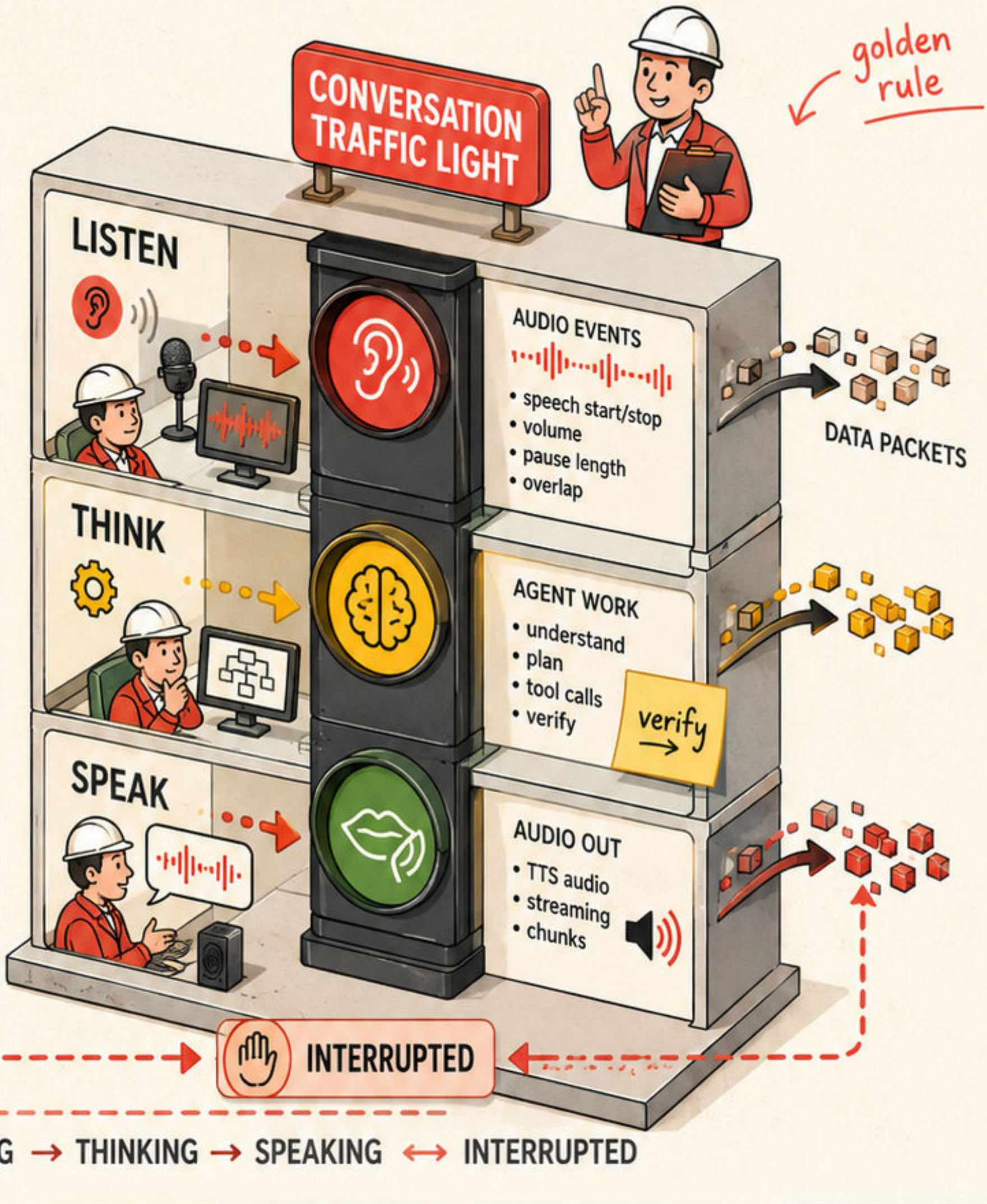
GOLDEN NOTES

Detect end of turn without long silence

Allow user interruption

Cancel unheard audio and stale tool plans

Log what the caller actually heard



CODE

```
state = "LISTENING"
def on_event(e):
    global state
    if is_user_interrupt(e):
        cancel_output()
        cancel_tool_plans()
        state = "LISTENING"
    else:
        handle_by_state(e)
```

INPUT

Audio timing events

OUTPUT

Natural turn transitions

COMMERCIAL LENS

Better turn-taking increases completion rate and customer satisfaction.

REPOSITORY

docs/architecture.md

TRANSLATION & DUBBING

Localization preserves meaning, timing, tone and cultural fit—not just words.

1

Transcribe with timestamps first

2

Translate for audience and context

3

Generate or cast voice with consent

4

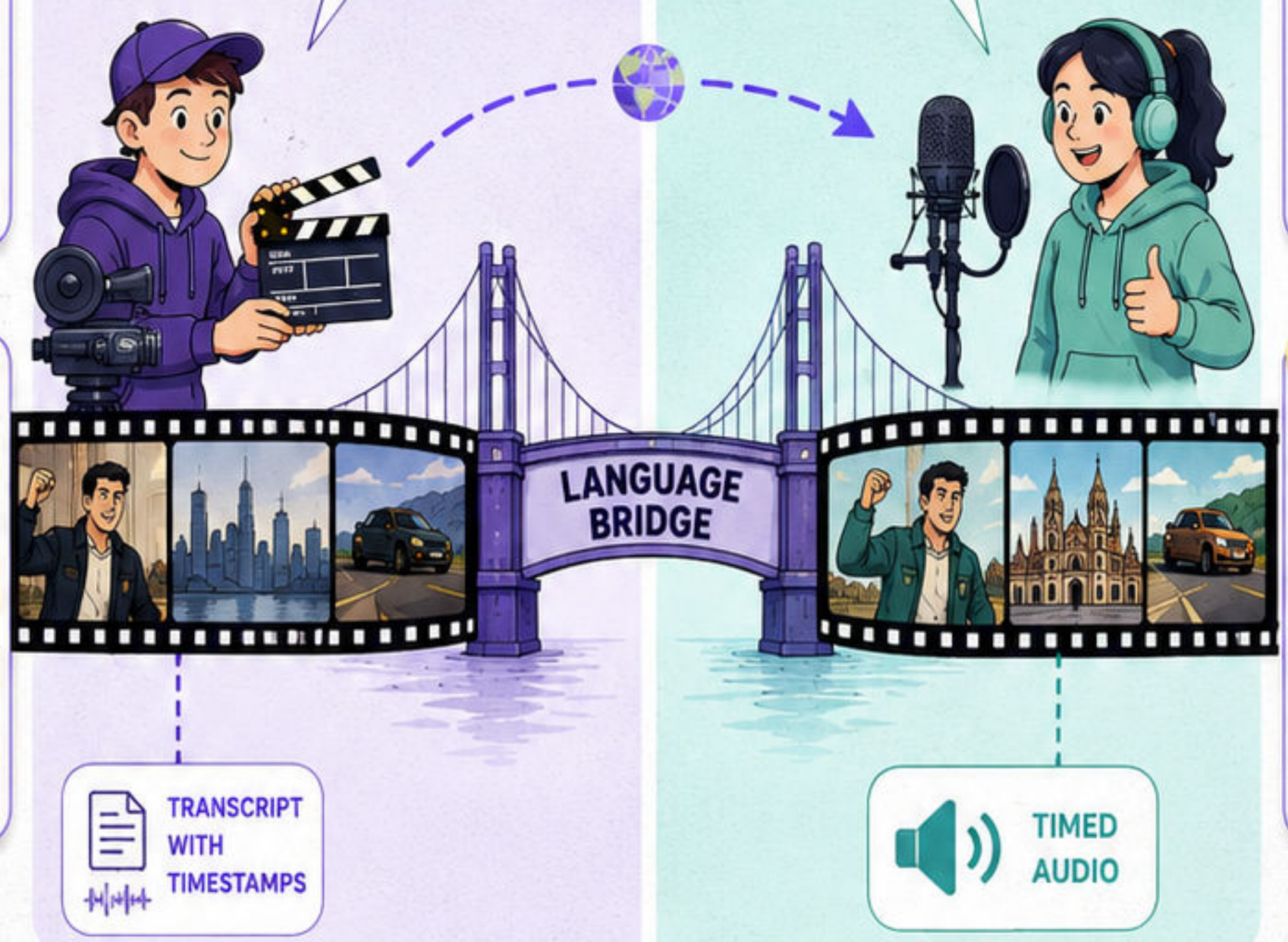
Review names, numbers, claims and lip timing

SOURCE (ORIGINAL)

00:00:02.840
We'll never give up.

TARGET (LOCALIZED)

00:00:02.840
Nunca nos daremos por vencidos.



TRANSCRIPT WITH TIMESTAMPS

TIMED AUDIO

golden rule

TRANSCRIPT → TRANSLATION → VOICE → MIX → REVIEW

verify

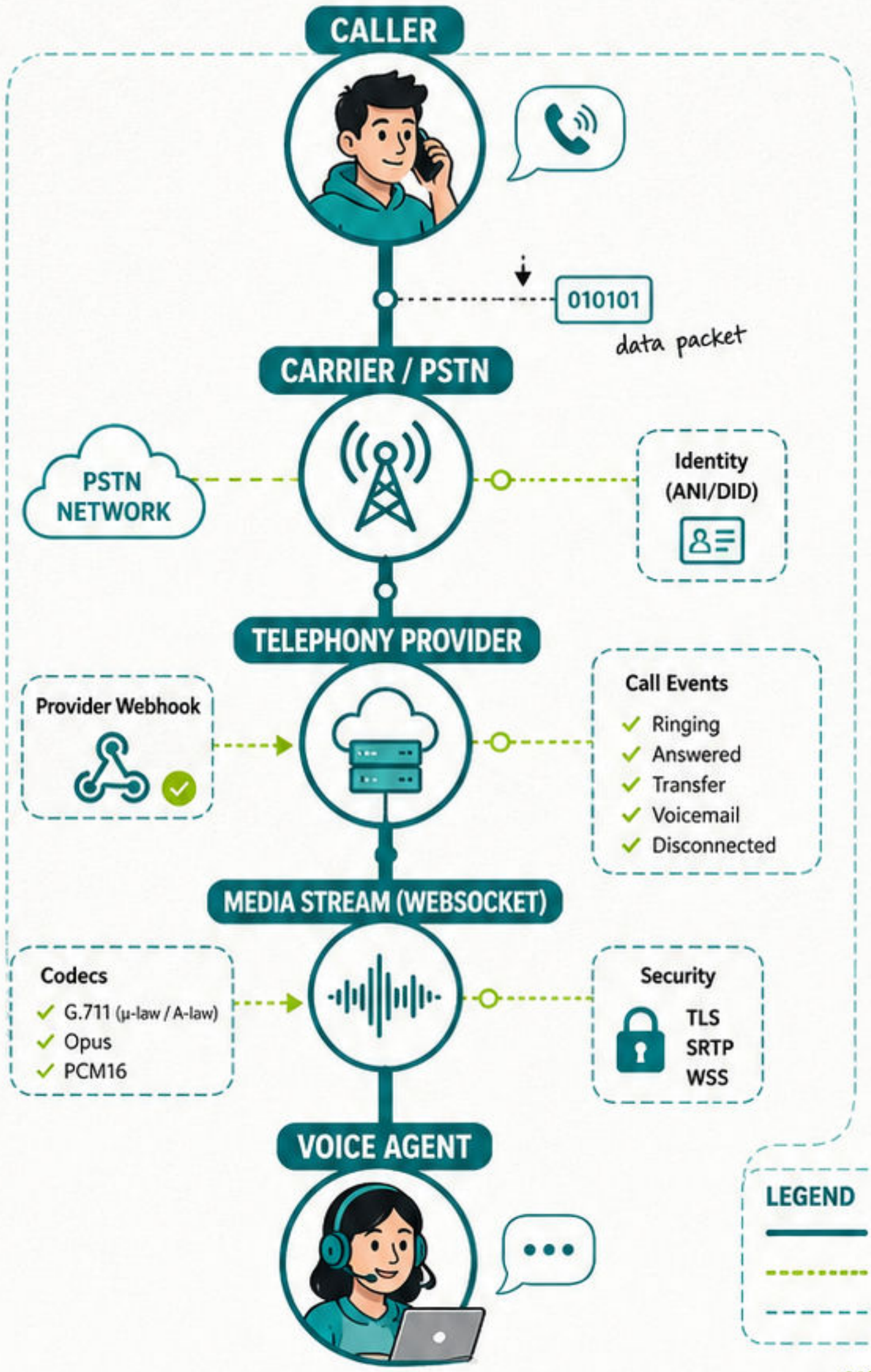
CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`audio → transcript → translation → voice → mix → review`	Authorized media + target locale	Timed localized media	Localization is a productized service with clear per-minute pricing. cost check	projects/blueprints.py

TELEPHONY INTEGRATION

A phone call adds networks, codecs, identity and operational risk.

GOLDEN NOTES

1. Validate provider webhooks and call identity
2. Convert audio formats deliberately
3. Handle transfer, voicemail and disconnects
4. Record only with lawful notice and consent



golden rule

cost check

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`PSTN → telephony provider → media stream → voice agent`	Inbound call	Resolution, transfer or safe fallback	Missed-call recovery has measurable commercial value.	`projects/blueprints.py`

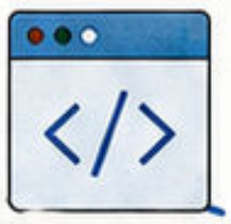
verify

VOICE SAFETY & CONSENT

A realistic voice is a capability with identity and fraud risk.

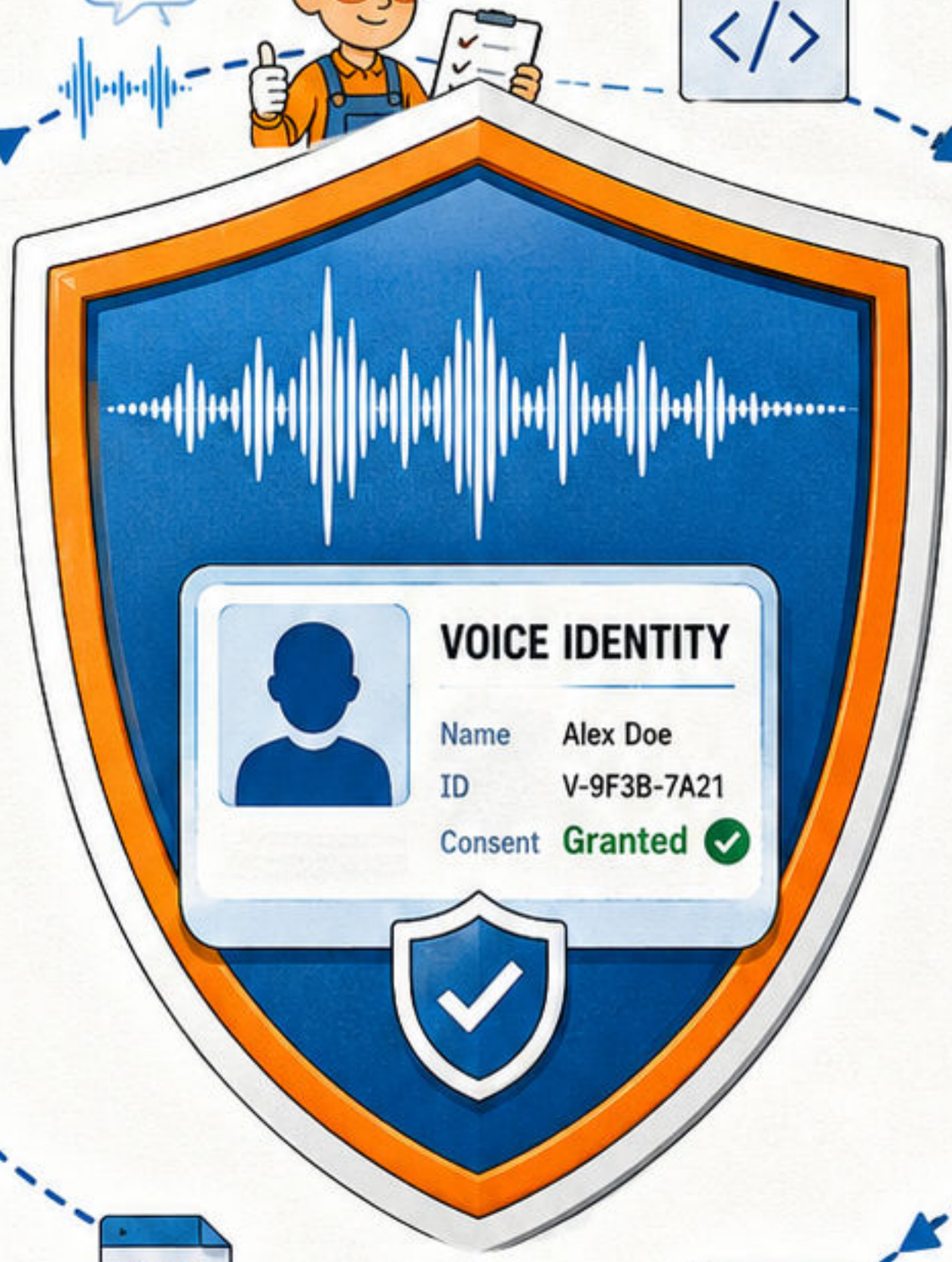
1

Obtain explicit permission for cloning



2

Prevent impersonation and deceptive use



3

Protect voice samples as sensitive data

4

Provide disclosure, revocation and audit trails

CODE

```
consent.valid &&
purpose.allowed
```

golden rule

INPUT

Voice sample + authorized purpose

OUTPUT

Bounded synthetic voice use

COMMERCIAL LENS

Trustworthy voice services can compete on governance.

cost check

REPOSITORY

SECURITY.md

VIDEO GENERATION APIS

Video is an expensive sequence of consistent frames, not a single image.

- Define shot, motion, duration and aspect ratio

- Use reference assets with rights

- Budget per second and per retry

- Review continuity, text, identity and claims

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`create_video(prompt, duration=5, ratio="16:9")`	Prompt + optional image/video references	Asynchronous video task	Short-form product packages can be sold per approved clip.	`examples/05_media/runway_video.py`

IMAGE TO VIDEO

**A strong first frame anchors identity;
motion instructions control the transformation.**

- Use a clean high-resolution source

- Describe camera and subject motion separately

- Avoid contradictory motion

- Compare first and last frames for drift

CODE

```
payload = {
  "image": IMAGE_PATH,
  "motion_prompt": MOTION_TEXT,
  "duration_sec": 5,
  "aspect_ratio": "16:9"
}
response = client.videos.create(payload)
job_id = response["id"]

print(job_id)
```

image + motion_prompt → video job

INPUT

Authorized image + motion brief

OUTPUT

Short generated clip

COMMERCIAL LENS

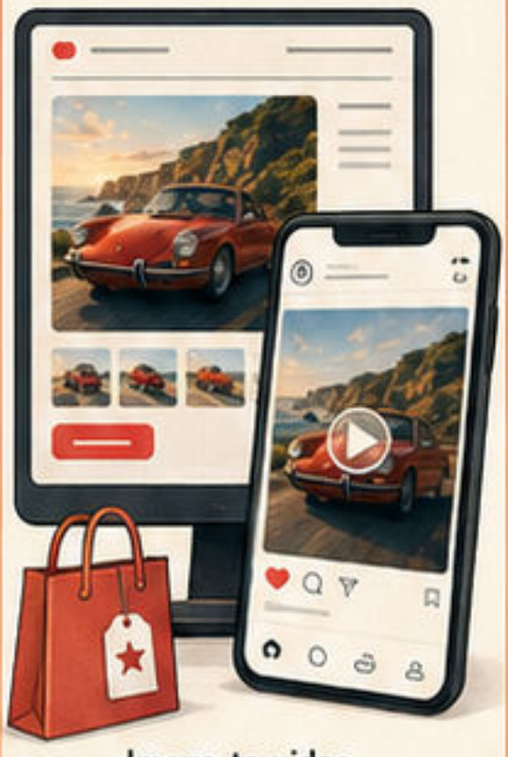


Image-to-video upgrades existing catalogs and campaigns.

REPOSITORY

examples/05_media/runway_video.py

VIDEO EDITING WITH AI

Editing should name the exact transformation and protected content.

Specify timestamps and intended change

Protect identity, product and brand details

EDITABLE SEGMENTS

AI may modify these parts.

LOCKED REGIONS

Protect identity, product and brand.

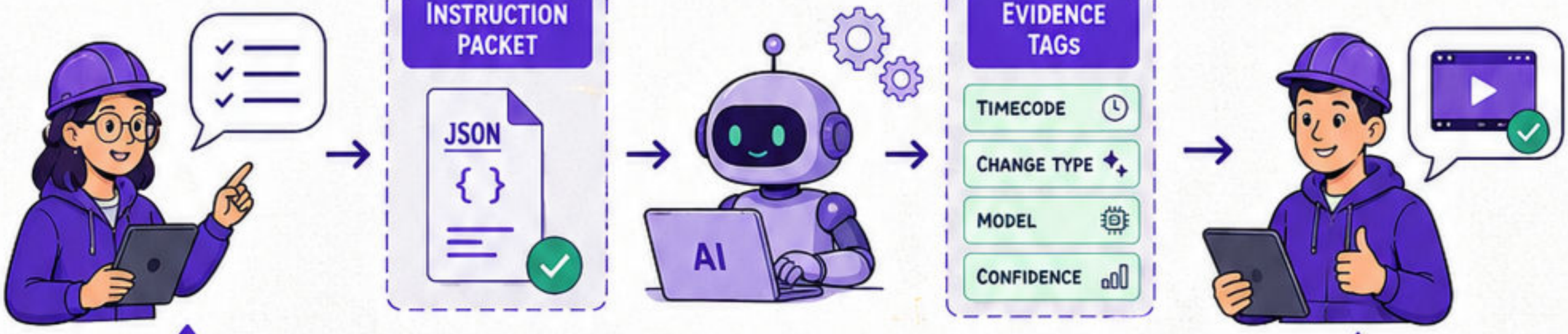
Replace background with sunset.

Keep person identity protected.

Add subtle motion and color grade.

Use keyframes when supported

Review every frame near transitions



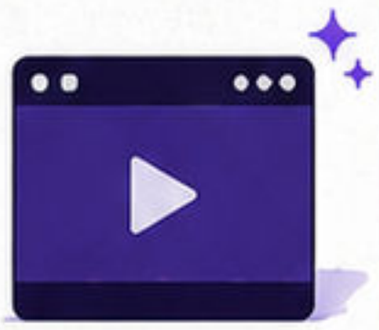
CODE

```
edit(video,
  instruction,
  keyframes)
```

INPUT

Video + edit boundary

OUTPUT



Edited video candidate

COMMERCIAL LENS

AI-assisted editing can reduce turnaround for media teams.

REPOSITORY

docs/provider-matrix.md

AVATARS & DIGITAL PRESENTERS

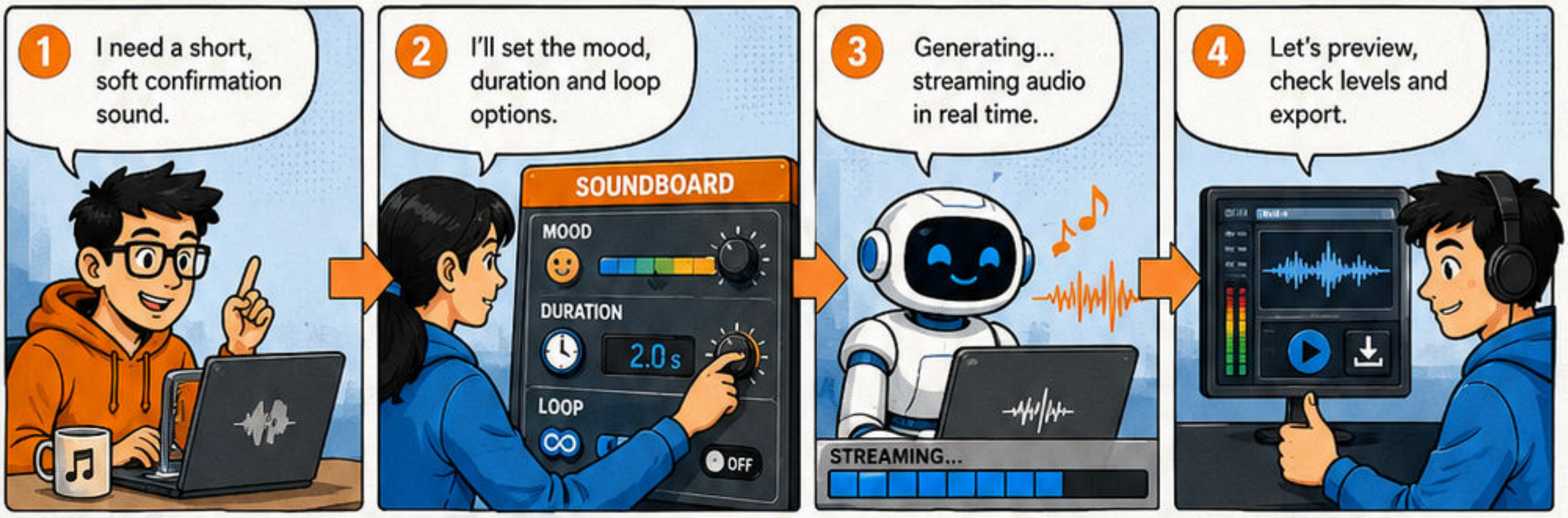
A presenter is a production layer—not permission to imitate a person.

- Use owned or licensed likenesses
- Bind avatar to approved scripts and knowledge
- Disclose synthetic representation
- Monitor identity drift and prohibited claims

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`payload = { "avatar": avatar_id, "approved_audio": audio_file, "disclosure": true } response = api.render( payload)`				
avatar + approved_audio + disclosure	Consented identity + script	Presenter video	Avatars support scalable training and multilingual explainers.	SECURITY.md

MUSIC & SOUND EFFECTS

Sound generation needs mood, timing, format and rights-aware review.



- Define purpose, duration, energy and instrumentation

- Avoid imitation requests involving living artists

- Generate stems or loops when workflow needs them

- Check loudness, clipping and usage rights

CODE

```
generate_sound(
"soft interface
 confirmation",
 seconds=2)
```

INPUT

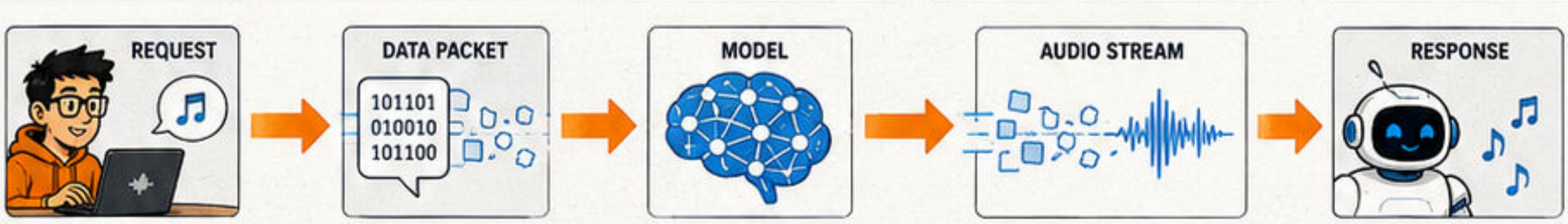
OUTPUT

COMMERCIAL LENS

Licensed sound packages can complement video and learning products.

REPOSITORY

docs/provider-matrix.md

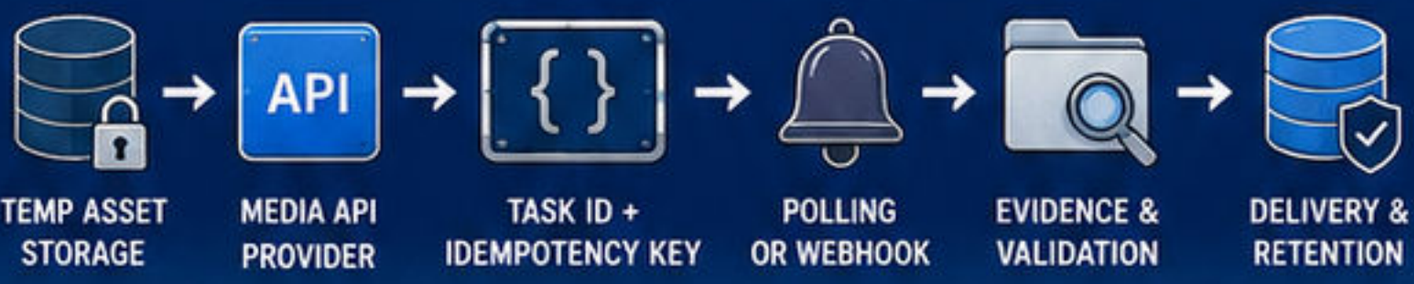


THE MEDIA JOB PIPELINE

Upload, create, monitor, validate, deliver and expire.

- Use secure temporary asset URLs
- Store provider task IDs and idempotency keys
- Queue polling and webhook work
- Expire outputs according to retention policy

DATA FLOW & EVIDENCE



STATUS BOARD

UPLOADED	12
CREATING	5
RUNNING	8
VALIDATED	7
DELIVERED	10
EXPIRED	3

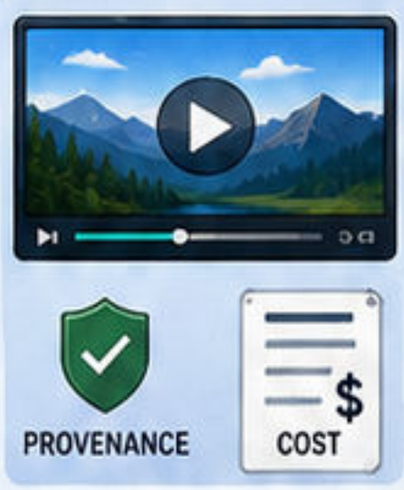
CODE

```
1 upload →
2 submit →
3 monitor →
4 validate →
5 deliver →
6 delete
```

INPUT

Media assets + brief

OUTPUT



Approved output + provenance + cost

COMMERCIAL LENS

Operational discipline protects margin in media products.

REPOSITORY

```
examples/
└ 05_media/
  └ runway_video.py
```

examples/05_media/runway_video.py

SECRETS & IDENTITY

Production authentication should be short-lived, scoped and observable.

01

Prefer workload identity where supported

02

Separate development, staging and production

03

Rotate and revoke automatically

04

Alert on unusual spend, region or request volume

CODE

```
credential =
secret_store.get("provider-key")
```

INPUT

Authenticated workload identity

OUTPUT

Scoped provider access

COMMERCIAL LENS

Enterprise buyers pay for governance and reliability.

REPOSITORY

SECURITY.md

PRIVACY & REDACTION

Collect less, retain less, expose less.

BEFORE: OVER-COLLECTING

Incoming Event (Raw)

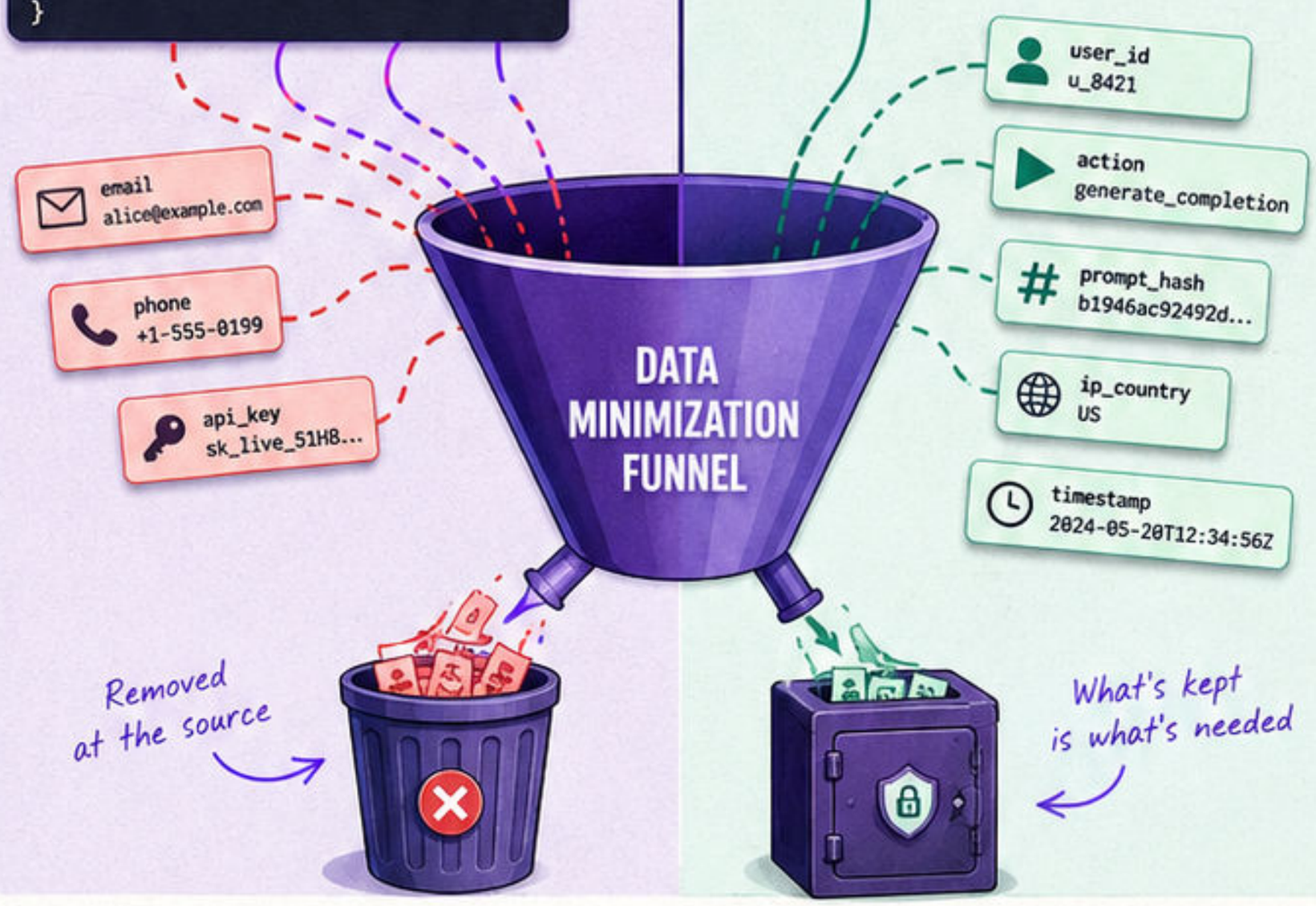
```
{
  "user_id": "u_8421",
  "email": "alice@example.com",
  "phone": "+1-555-0199",
  "api_key": "sk_live_51H8...",
  "action": "generate_completion",
  "prompt": "Summarize Q3 results",
  "ip": "203.0.113.7",
  "timestamp": "2024-05-20T12:34:56Z"
}
```

AFTER: DATA MINIMIZATION

Minimized Event (Safe)

```
{
  "user_id": "u_8421",
  "action": "generate_completion",
  "prompt_hash": "b1946ac92492d...",
  "ip_country": "US",
  "timestamp": "2024-05-20T12:34:56Z"
}
```

- Classify data before transmission
- Redact PII and credentials from logs
- Choose region and retention deliberately
- Support deletion and access workflows



CODE

```
def redact_mapping(event):
  mapping = {
    "email": "[REDACTED]",
    "phone": "[REDACTED]",
    "api_key": "[REDACTED]"
  }
  return apply_mapping(event,
                       mapping)
```

`safe_log = redact_mapping(event)`

INPUT

Request metadata with sensitive fields

OUTPUT

Useful telemetry without exposed secrets

COMMERCIAL LENS

Privacy-aware design expands the customers you can serve.

REPOSITORY

src/ai_api_playbook/safety.py

RATE LIMITS & CONCURRENCY

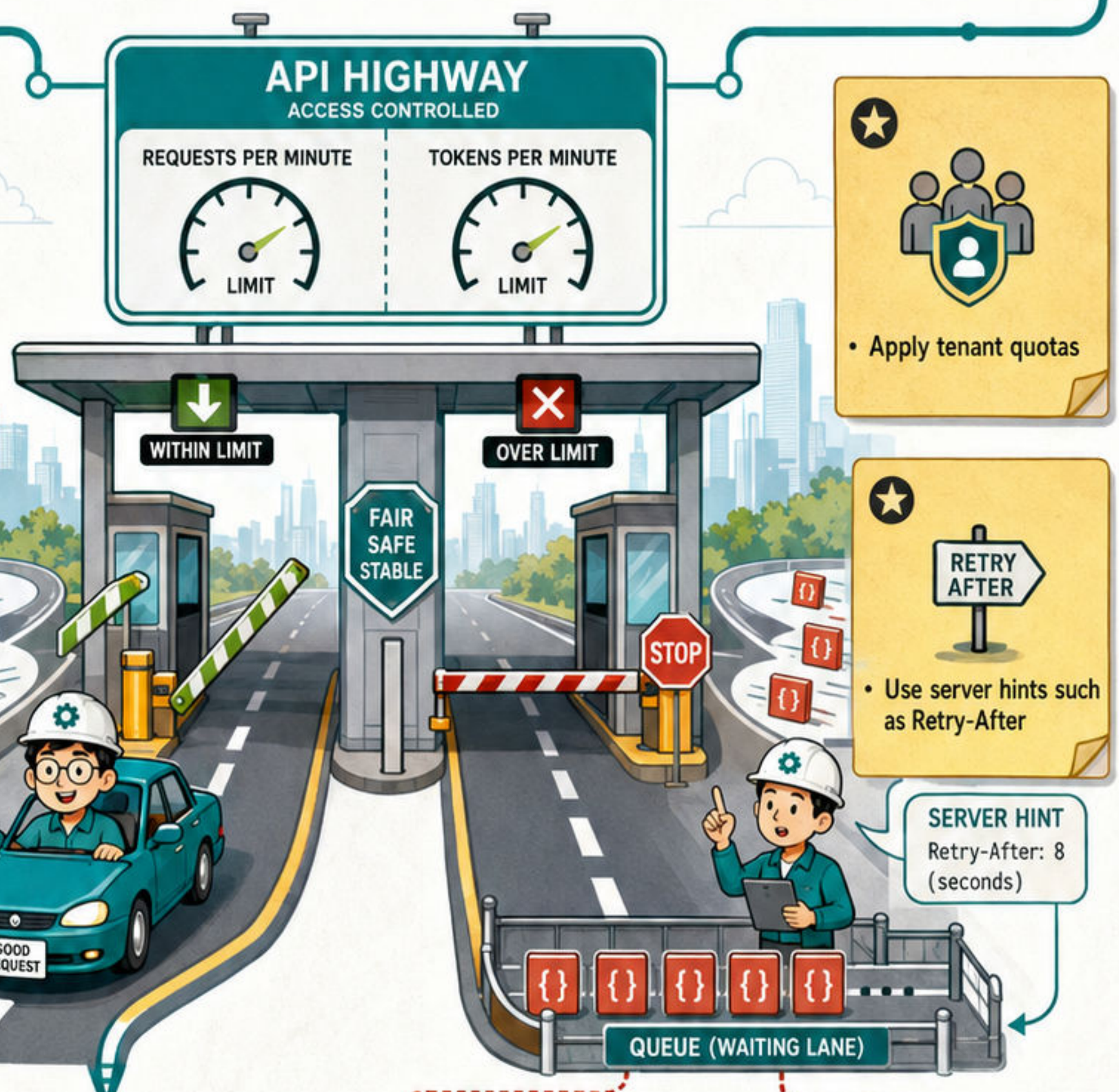
Throughput is a budgeted resource, not an invitation to retry everything at once.

- Track request, token and concurrent-job limits

- Apply tenant quotas

- Queue excess work

- Use server hints such as Retry-After



PROCESSED

Delivered successfully

DISPATCHER & SCHEDULER

- Respect limits
- Fair to all tenants
- Maximize throughput
- Avoid retries storms

WILL RETRY LATER

Resubmitted after Retry-After

CODE

```
semaphore =
Limit(concurrency=5)
```

INPUT

Burst of jobs

OUTPUT

Smoothed queue with predictable throughput

COMMERCIAL LENS

Capacity control prevents outages and surprise bills.

REPOSITORY

docs/architecture.md

RETRY & BACKOFF

Retry transient failures with patience, jitter and a stopping rule.

1

Retry timeouts, connection failures and selected 5xx/429 errors

2

Do not blindly retry validation or authorization failures

Step wider when the path is rough.

ATTEMPT 5
WAIT ~16s

ATTEMPT 4
WAIT ~8s

ATTEMPT 3
WAIT ~4s

ATTEMPT 2
WAIT ~2s

ATTEMPT 1
. WAIT ~1s

SUCCESS

3

Use exponential backoff with jitter

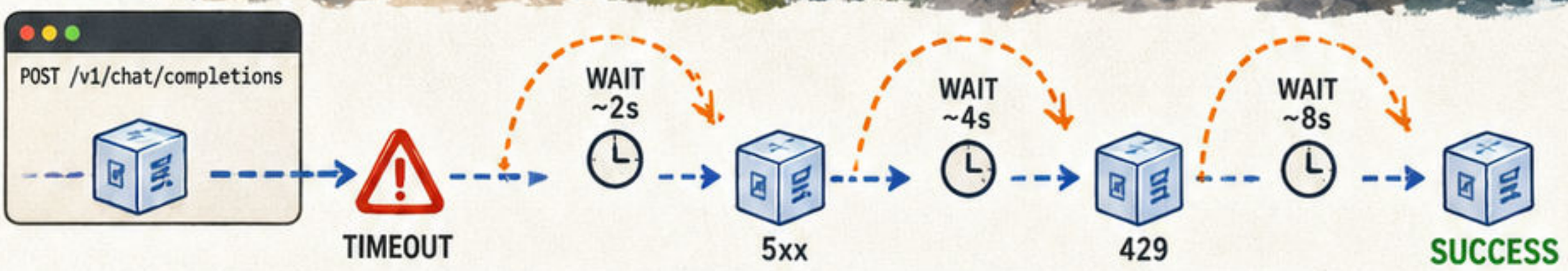
STOPPING RULES

- Max attempts reached
- Deadline exceeded
- Non-retryable error

4

Protect billable writes with idempotency

Fail fast on permanent errors.



CODE

```
delay = random(0,
        min(cap,
       base*2**attempt))
```

INPUT

Transient failure

OUTPUT

Recovered result or controlled final error

COMMERCIAL LENS

Correct retry logic improves reliability without multiplying cost.

REPOSITORY

`src/ai_api_playbook/retry.py`

CACHING

Pay once, Reuse many!

Do not pay repeatedly for the same stable work.

1. Request comes in

DATA PACKET

```
{}  10101010
    01010101
    10101010
```

2. Compute cache key

KEY

a7f3e9b2c1d4...

- Cache only when inputs and policy match
- Hash normalized prompts and source versions

- Set TTL by data freshness
- Never leak cached results across tenants

4. Return instantly

CACHE HIT!
Serve from fridge

CACHE MISS
Go to model/API

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`key = hash(model + prompt + source_version + policy)`	Repeated deterministic request	Cached validated response	Caching directly improves speed and gross margin.	docs/architecture.md

COST CONTROL

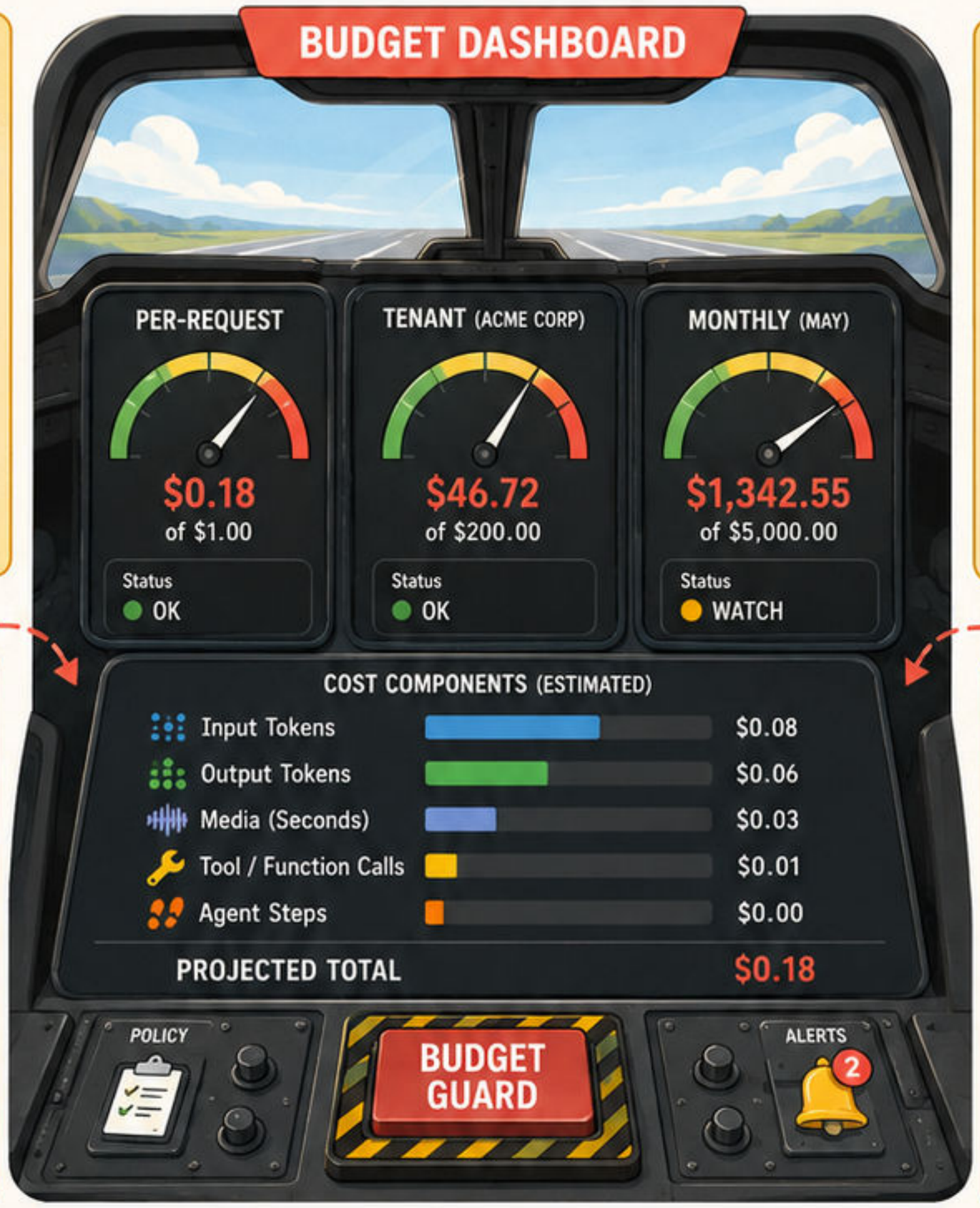
Every request needs a maximum acceptable cost before it starts.

GOLDEN NOTE 1

- Estimate input before calling

GOLDEN NOTE 2

- Cap output and agent steps



GOLDEN NOTE 3

- Track media seconds and tool charges

GOLDEN NOTE 4

- Stop or downgrade only according to product policy

CODE

```
if projected_cost > budget:
    raise BudgetExceededError
```

INPUT

PRICE TABLE (USD)		
ITEM	UNIT	PRICE
Input Token	1K tokens	$0.0015
Output Token	1K tokens	$0.0060
Audio (STT)	1 sec	$0.0060
Audio (TTS)	1 sec	$0.0120
Tool Call	per call	$0.0400

Planned request + price table

OUTPUT

APPROVED ROUTE

or

BUDGET REJECTION

Approved route or budget rejection

COMMERCIAL LENS

Cost-aware architecture makes revenue sustainable.

REPOSITORY

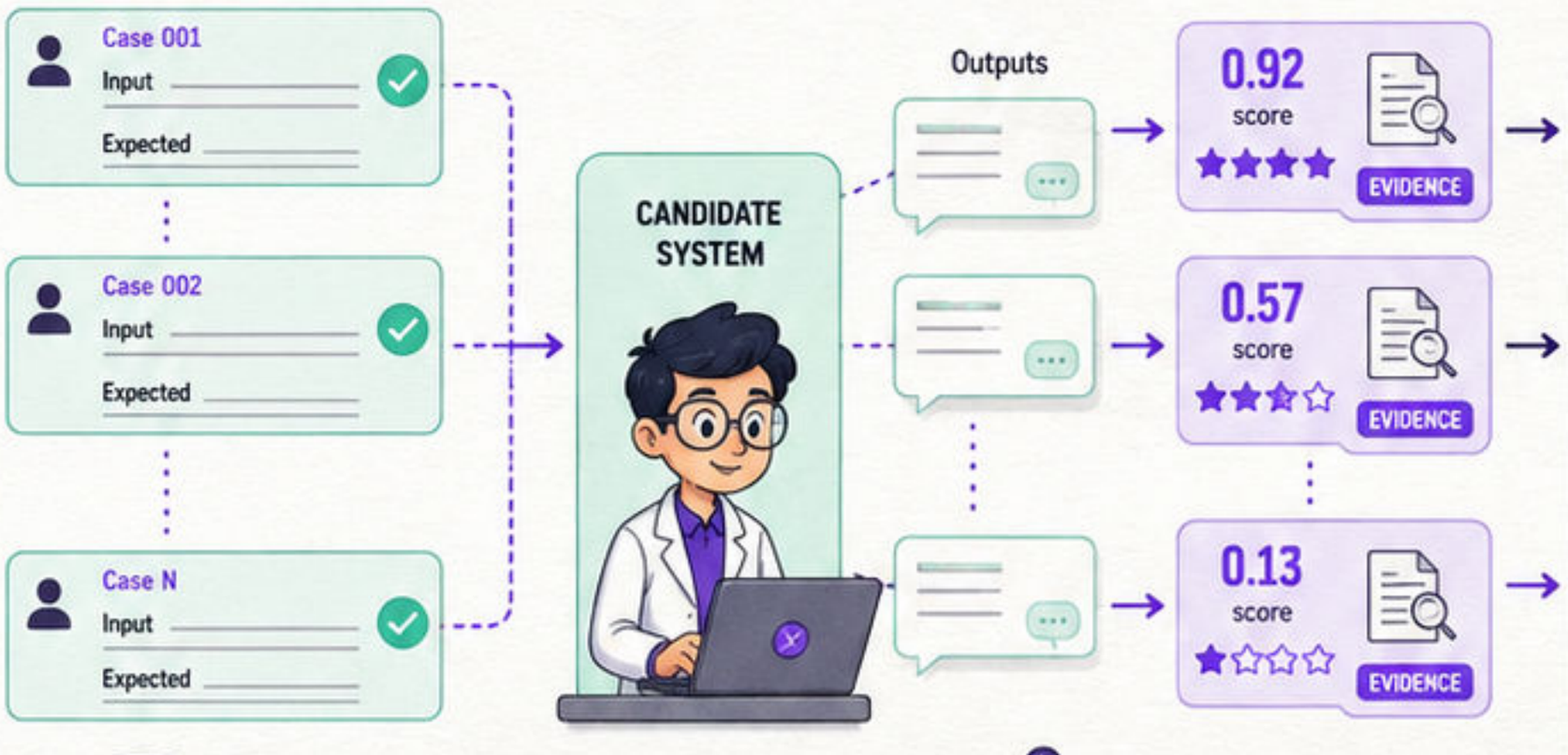
src/ai_api_playbook/economics.py

EVALUATION

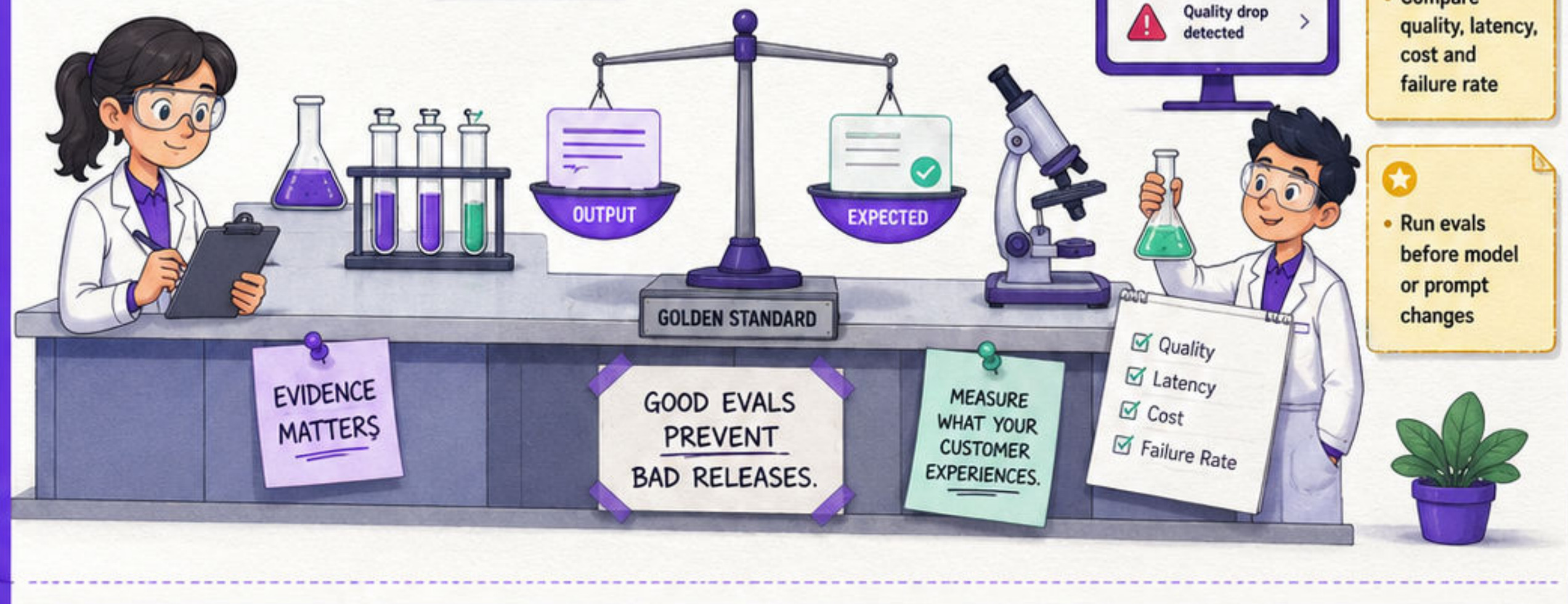
Measure the behavior your customer buys.

1. TEST CASES → **2. RUN CANDIDATE** → **3. EVALUATE** → **4. RESULTS**

Golden cases (inputs + expected) | Your system (model or prompt) | Score against expected with evidence | Aggregate metrics and alerts



- Build representative test cases
- Define task-specific rubrics
- Compare quality, latency, cost and failure rate
- Run evals before model or prompt changes



CODE

```
score = evaluate(
  output,
  expected,
  evidence
)
```

INPUT

Test dataset + candidate system

OUTPUT

Comparable metrics and regression alerts

COMMERCIAL LENS

Evaluation enables confident upgrades and premium guarantees.

REPOSITORY

```
src/
  ai_api_playbook/
    evaluation.py
```

OBSERVABILITY

Trace one customer outcome across models, tools, retrieval and validation.

1

- Use correlation IDs across services

2

- Record safe inputs, route, latency, usage and status

3

- Separate provider failure from product failure

4

- Create alerts on success rate and cost per successful job

CUSTOMER OUTCOME

Job completed successfully

VALIDATION SPAN

verify output

EVIDENCE
schema valid
policy pass
no PII leaked

RETRIEVAL SPAN

find context

EVIDENCE
chunk count
sources: 7
score ≥ 0.78

MODEL SPAN

generate answer

TOOL SPAN

call tool

GPT-4o
1.8s | $0.012
200 tokens

Claude 3.5
2.2s | $0.015
180 tokens

Calculator
0.6s | $0.001
status: ok

WebSearch
1.4s | $0.003
status: ok

REQUEST RECEIVED

user_id: 12345 correlation_id: 8f3a-7c2d-91b4
request: answer user question

Span (event) ----> Flow Evidence Metric System

CODE

```
trace_id →
model span →
tool span →
validation span
```

INPUT

Events from the full workflow

OUTPUT

Metrics, traces and actionable alerts

COMMERCIAL LENS

Operational visibility reduces downtime and support burden.

REPOSITORY

docs/architecture.md

PROJECT 1 — CITED KNOWLEDGE ASSISTANT

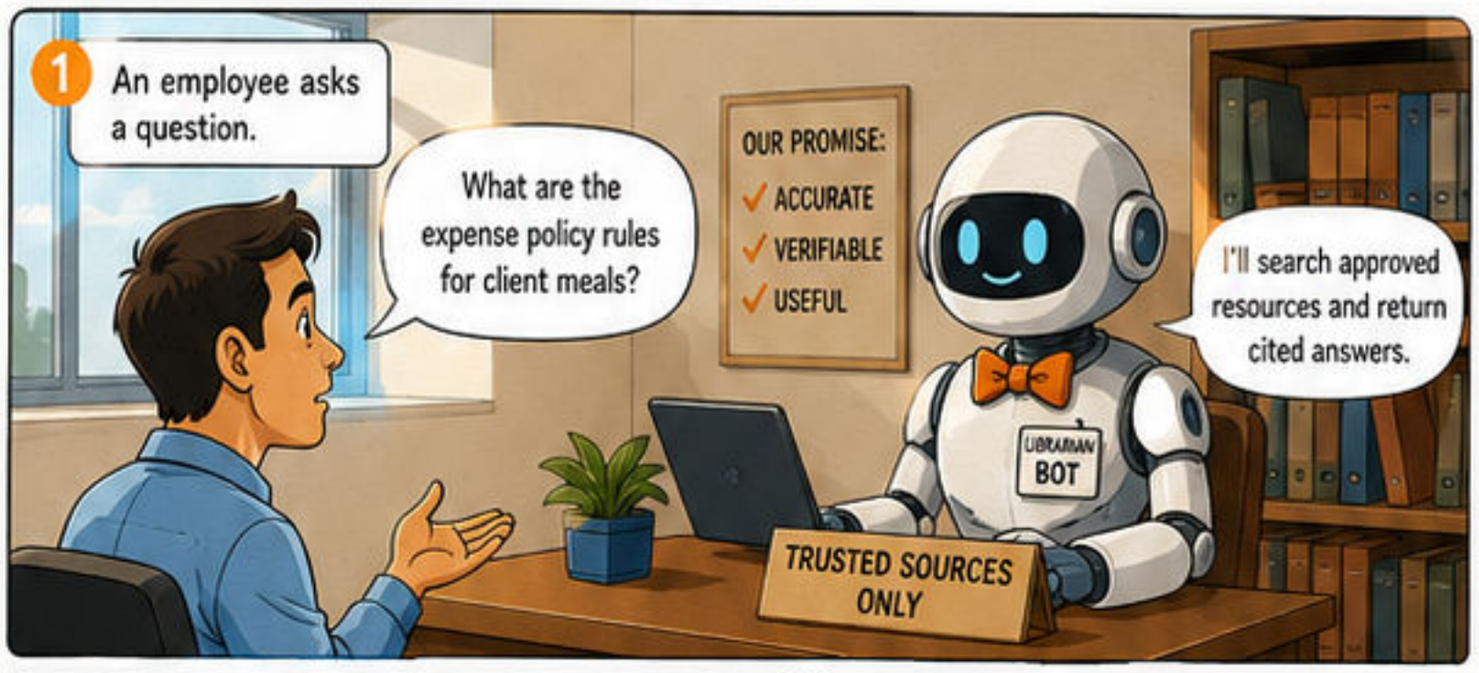
Turn approved documents into answers employees can verify.

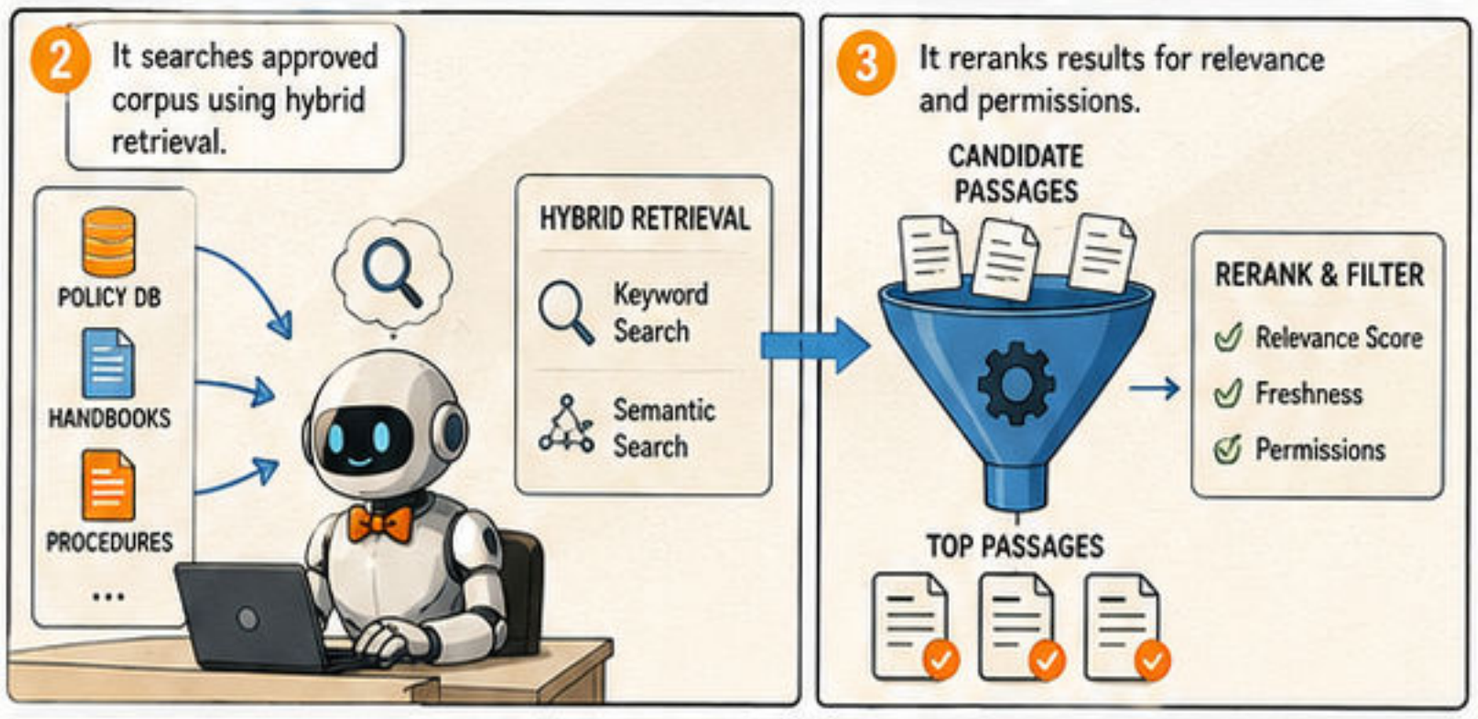
BUYER

- Buyer: SME operations team

CORE STACK

- Core stack: ingestion, hybrid retrieval, reranking, grounded generation



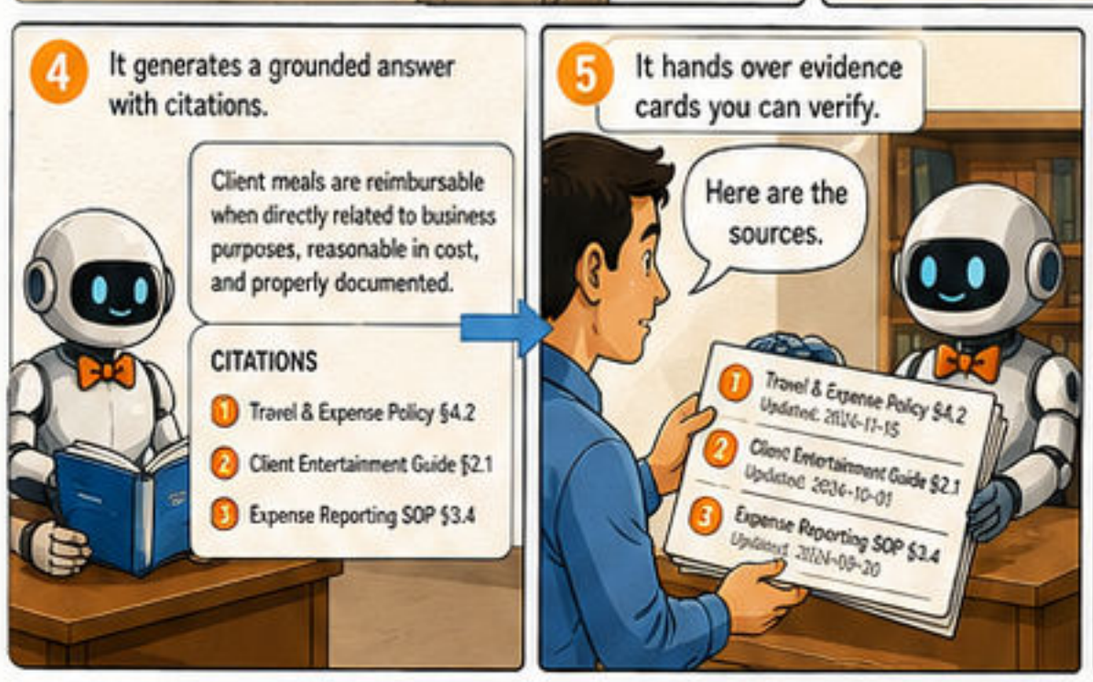


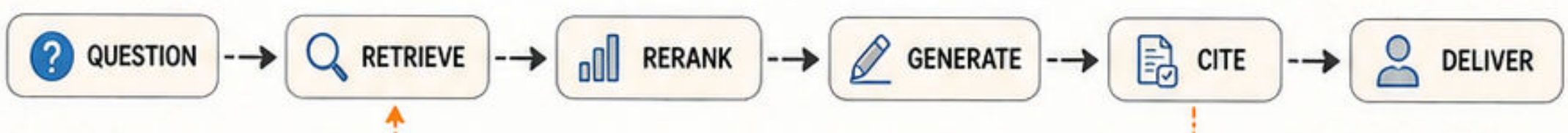
PAIN

- Pain: repeated policy search

SUCCESS

- Success: useful answer with correct source IDs in under 15 seconds





CODE

```
answer = rag(question,
       tenant, permissions)
```

INPUT

Question + authorized corpus

OUTPUT

Answer + citations + abstention path

COMMERCIAL LENS

Charge setup + monthly subscription + usage.

REPOSITORY

projects/blueprints.py

KNOWLEDGE ASSISTANT – ECONOMICS

Sell verified time saved, not chatbot messages.

01

- Measure answer success rate

02

- Calculate cost per successful answer

UNIT ECONOMICS SEE-SAW

Sustainable when value (left) exceeds total cost (right).

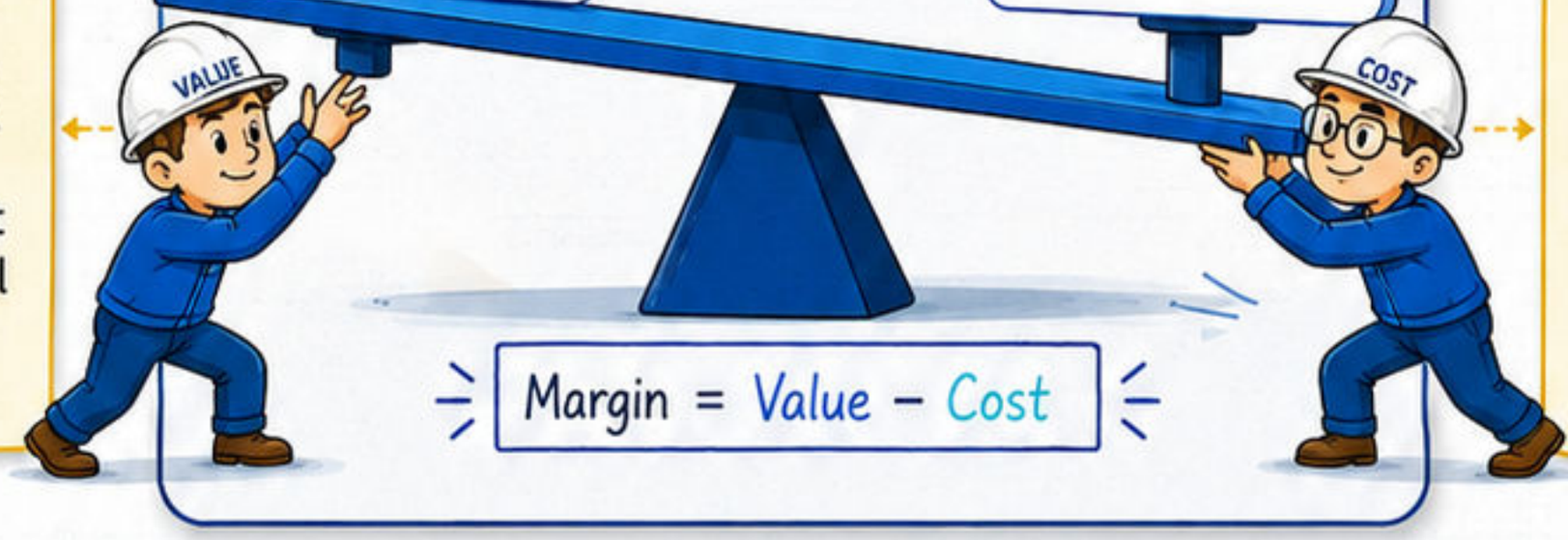
SUBSCRIPTION VALUE

- Time Saved (Hours)
- Willingness to Pay
- Verified Success Rate

GROSS MARGIN

VARIABLE COST

- API Costs (Per Answer)
- Infra Costs (Compute, Storage)
- Support Costs (Per Case)



Margin = Value – Cost

03

- Create a human escalation queue

04

- Protect tenant isolation and document permissions

USAGE & EVENTS

MEASUREMENT & ATTRIBUTION

ECONOMIC OUTCOME

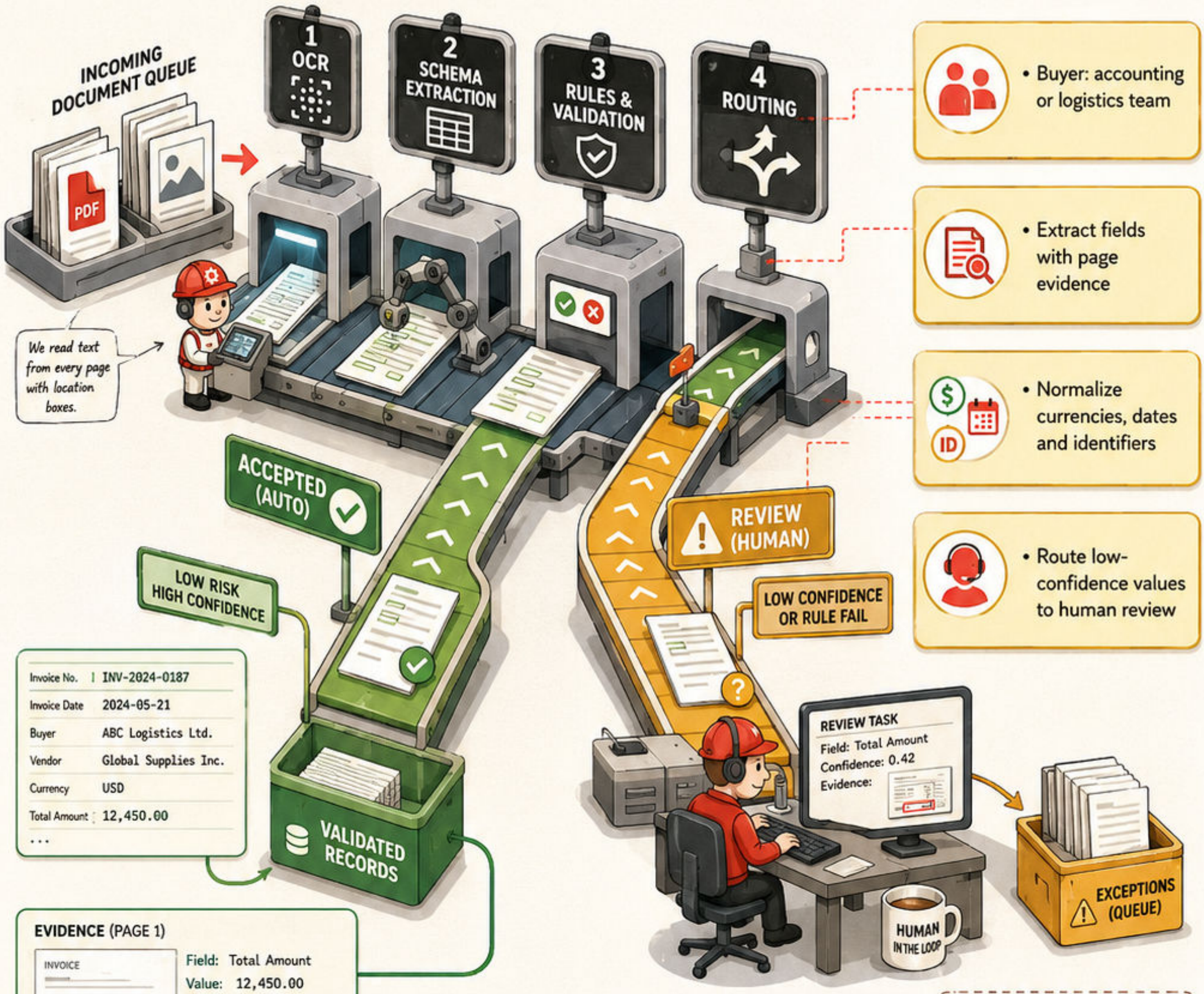
GROSS MARGIN
(% of Revenue)

Improve what you measure. Measure what drives margin.

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`margin = price - api - infra - support`	Usage, success and support metrics	Gross margin and improvement plan	Recurring knowledge value creates sticky subscriptions.	`src/ai_api_playbook/economics.py`

PROJECT 2 – DOCUMENT INTELLIGENCE

Convert document queues into validated records and exceptions.



CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`document → OCR → schema → rules → review queue`	PDFs and scans	Validated records + exceptions	Price per verified document plus premium review tier.	`$ projects/ > blueprints.py`

DOCUMENT INTELLIGENCE — QUALITY LOOP

Accuracy is a workflow of extraction, rules and review.

DOCUMENT INPUT

- Scanned PDFs
- Images
- Docs
- ... More formats

1 MODEL (EXTRACT)

Field candidates

- Vendor: ACME Inc.
- Invoice #: INV-3921
- Date: 2024-04-16
- Total: $1,254.00
- ...

1

- Create field-level ground truth

2

- Track straight-through processing rate

2 VALIDATOR (RULES)

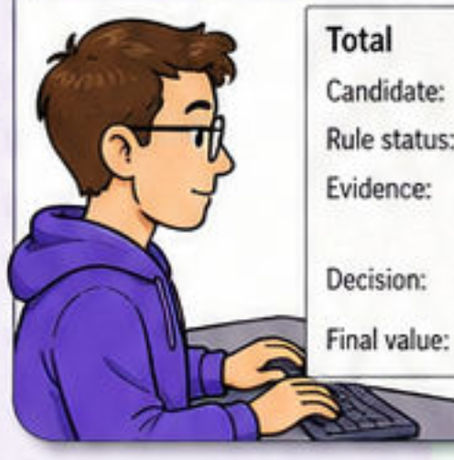
- Vendor: ACME Inc. ✓
- Invoice #: INV-3921 ✓
- Date: 2024-04-16 ✓
- Total: $1,254.00 ⚠
- ...

Rules, formats, lookups, business constraints

3

- Analyze errors by document type

3 REVIEWER (AUDIT)



Total
Candidate: $1,254.00
Rule status: Failed
Evidence: Total $1,254.00
Decision: Correct value
Final value: $1,254.00

Needs review

Decision & evidence captured

4

- Never hide uncertainty behind one document-level score

Improve continuously

4 FEEDBACK (LEARN)

- Labeled fields
- Error patterns
- Model updates

Better training data, better rules

QUALITY FLYWHEEL

Better data → better model → fewer errors → faster throughput

CODE

```
field = {
  value,
  evidence,
  confidence,
  status
}
```

INPUT

Extracted candidates

OUTPUT

Field	Value	Evidence	Conf.	Status
Vendor	ACME Inc.		0.98	✓
Invoice #	INV-3921		0.99	✓
Date	2024-04-16		0.97	✓
Total	$1,254.00		0.62	⚠
...	...	...	...	...

Audited field dataset

COMMERCIAL LENS

Operational data becomes a defensible improvement asset.

REPOSITORY

projects/publishing_engine/pipeline.py

PROJECT 3 — VOICE RECEPTIONIST

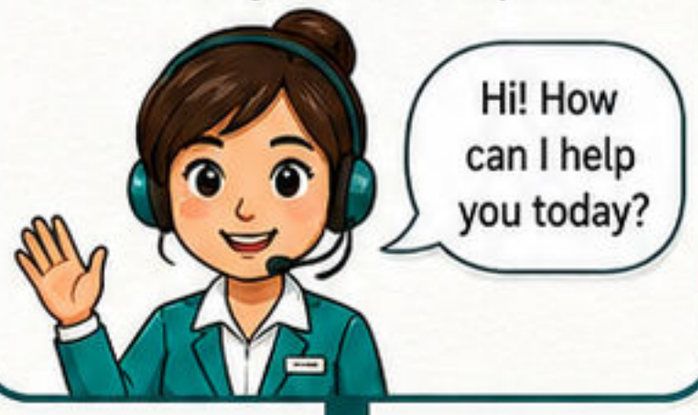
Recover missed opportunities without pretending uncertainty is certainty.

GOLDEN NOTE 1

1 • Buyer: clinics and local services

VIRTUAL RECEPTIONIST

Friendly. Clear. Helpful.



Live Voice Stream

10101
01010
10101

Data Packets

GOLDEN NOTE 2

2 • Connect telephony, realtime voice and scheduling tools

GOLDEN NOTE 3

3 • Confirm names, dates and contact details

GOLDEN NOTE 4

4 • Transfer sensitive or unclear calls

BOOKING

Capture details and schedule.

ANSWER

Answer questions with confidence.

HUMAN TRANSFER

Warm transfer to a human agent.

INBOUND CALL

Caller dials in.

VOICE CAPTURE

Realtime audio is captured.

UNDERSTAND

Extract intent and key information.

ROUTE

Decide: book, answer or transfer.

CONFIRM & LOG

Confirm outcome and log summary.

STORE & SYNC

Save data and sync with tools.

BOOKING PATH · ANSWER PATH · HUMAN TRANSFER PATH · DATA PACKETS (Secure & Encrypted)

CODE

```
def receptionist(call):
  intent, data = identify(call)
  if intent == "book":
    return book(data)
  if intent == "answer":
    return answer(data)
  return transfer_human(call)

call → identify → resolve |
  book | transfer
```

INPUT

Inbound call

OUTPUT

Confirmed outcome + summary

COMMERCIAL LENS

Charge setup, monthly platform fee and included minutes.

REPOSITORY

projects/blueprints.py

VOICE RECEPTIONIST — SAFETY & KPI

Completion matters only when consent, accuracy and escalation work.

1

- Track resolved-call rate

2

- Measure booking correctness

VOICE RECEPTIONIST DASHBOARD

COMPLETION RATE

78%

Resolved / Eligible Calls

TRANSFER RATE

12%

Escalated / Eligible Calls

LATENCY (TIME TO FIRST AUDIO)

1.4s

Good ≤ 2.0s

CONSENT RATE

95%

Explicit Consent Captured

RECENT CALL ACTIVITY — Success · Transfer · Failure

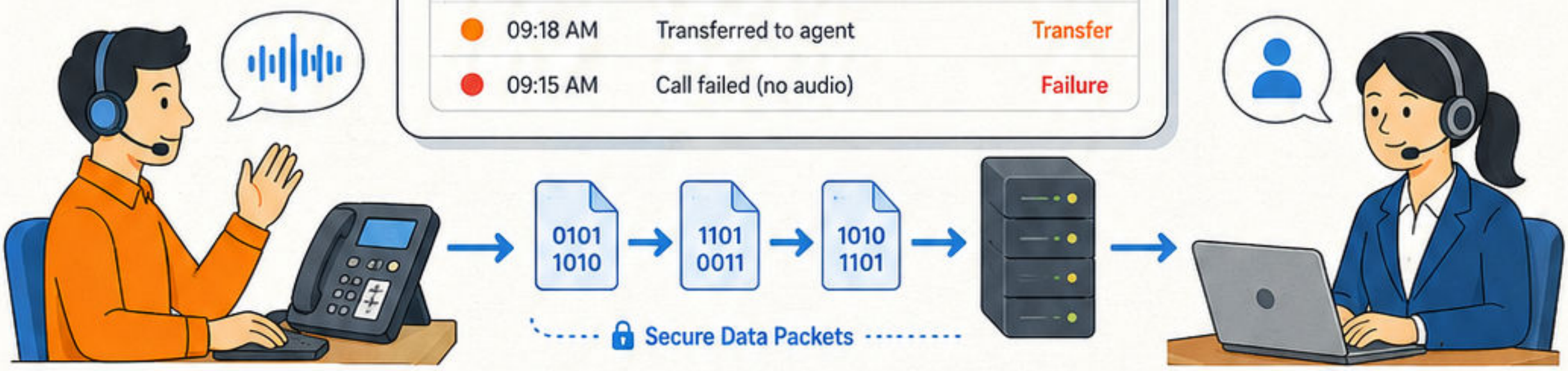
09:21 AM	Booking confirmed	Success
09:18 AM	Transferred to agent	Transfer
09:15 AM	Call failed (no audio)	Failure

3

- Record time to first audio and interruption quality

4

- Audit consent, transfer and failure behavior



CODE

```
# KPI formula
KPI =
  successful_verified_outcomes
  / eligible_calls
```

INPUT

Call events and outcomes

- Call start / end
- Audio latency
- Booking result
- Transfer events
- Consent captured
- Failures / errors

OUTPUT

Operational and safety dashboard

COMMERCIAL LENS

Reliability justifies higher-value B2B pricing.

REPOSITORY

projects/blueprints.py

PROJECT 4 — E-COMMERCE CATALOG ENGINE

Transform approved product data into consistent multilingual sales assets.

GOLDEN NOTES

1

Buyer: retailers and exporters

2

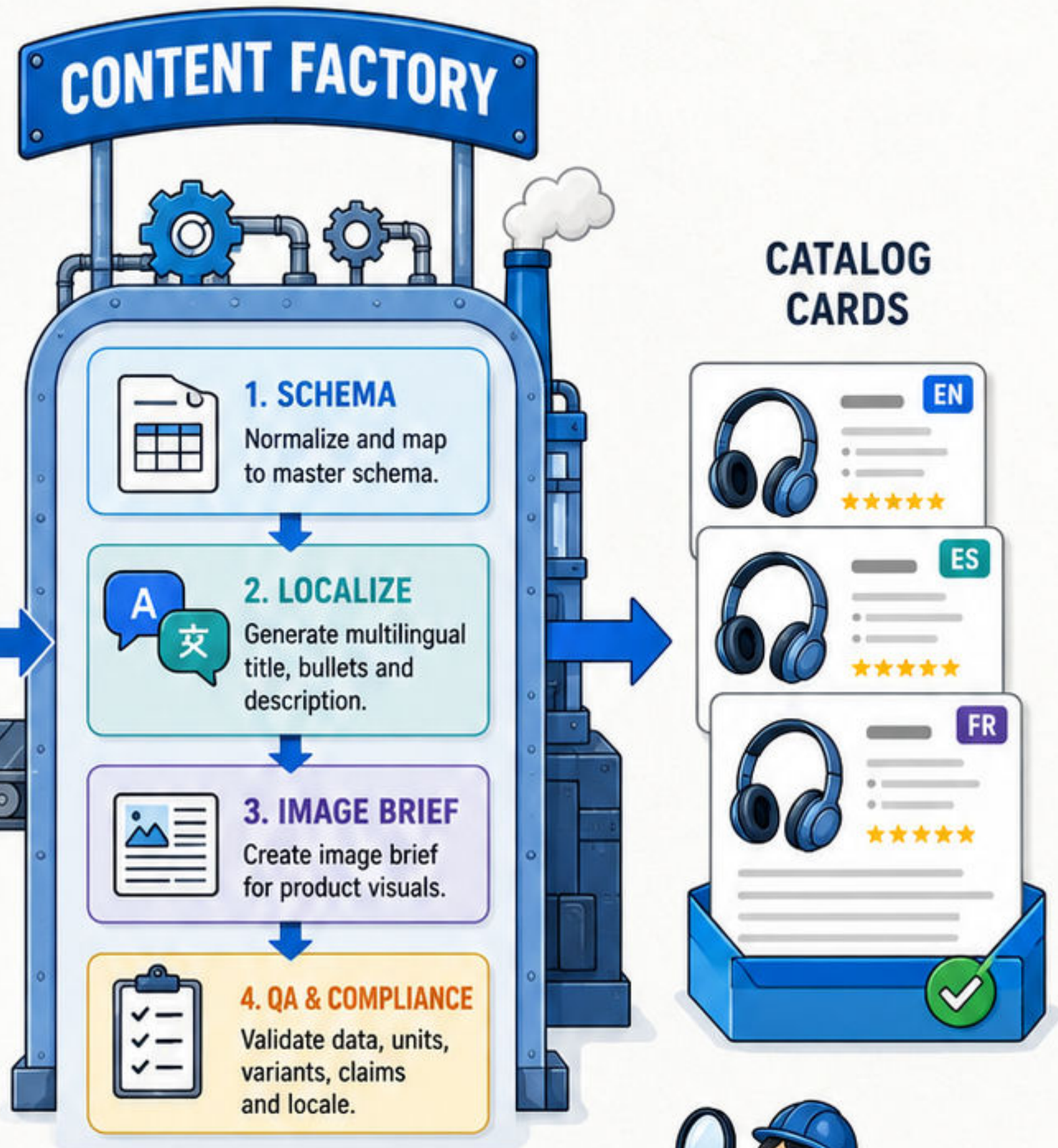
Ground every claim in product data

3

Generate title, bullets, description and image brief

4

Validate units, variants, prohibited claims and locale



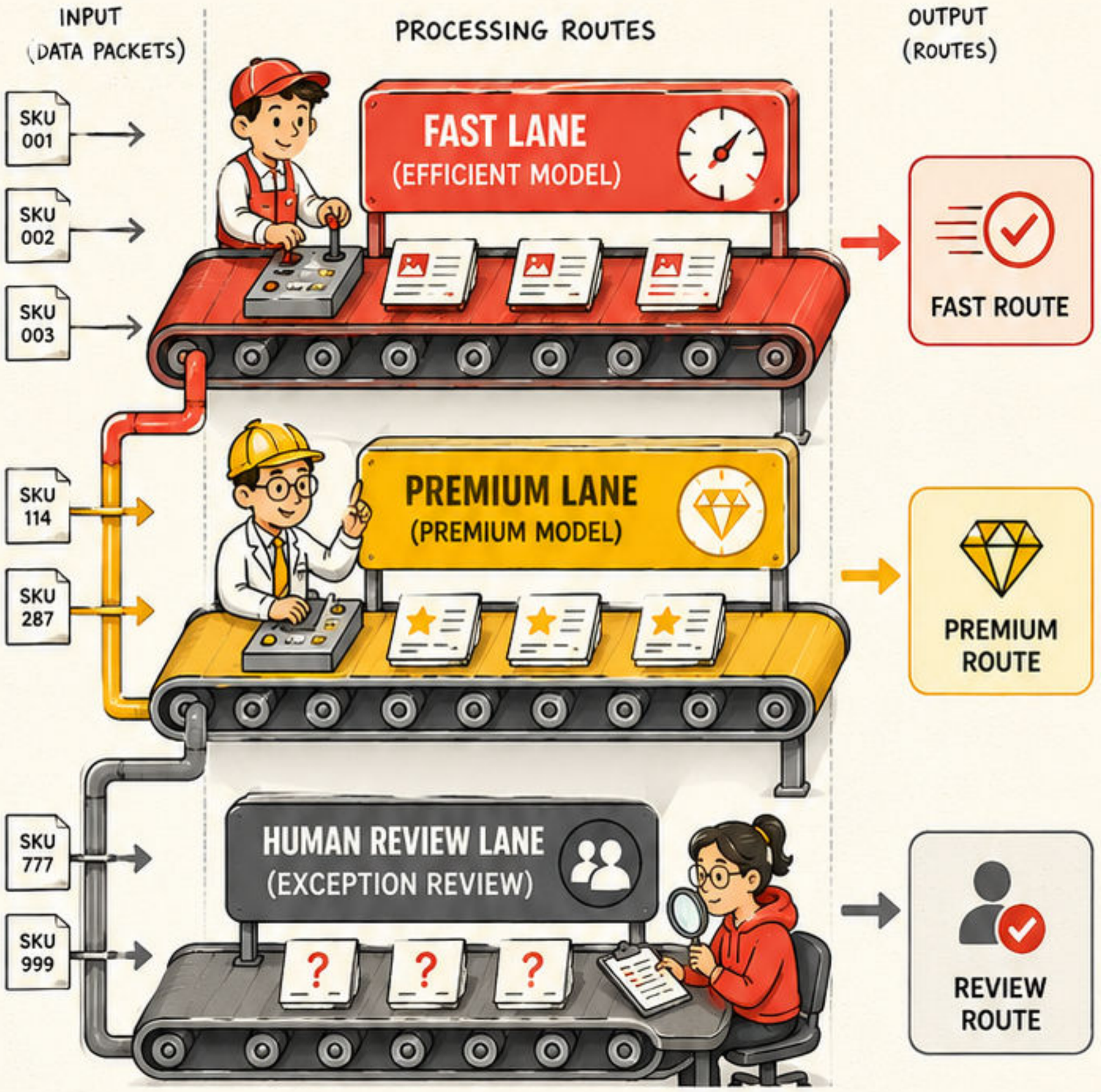
CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY

`def pipeline(sku_data):`
`  schema = normalize(sku_data)`
`  localized = localize(schema)`
`  pack = build_pack(localized)`
`  qa(pack)`
`  return pack`
sku_data → schema → localized copy → QA
	Structured SKU + images	Approved channel-ready product pack	Price per SKU, batch or monthly catalog volume.	projects/ blueprints.py

CATALOG ENGINE — MARGIN DESIGN

Batch stable work; reserve premium models for difficult exceptions.

1. Classify SKU complexity first
2. Cache brand and policy context
3. Batch routine generations
4. Measure approval rate and revisions per SKU



CODE

```
route = premium if
complexity > threshold
else efficient
```

INPUT

SKU complexity + policy

OUTPUT

Cost-controlled production route

COMMERCIAL LENS

Operational routing protects margins at catalog scale.

golden rule

REPOSITORY

src/ai_api_playbook/routing.py

verify

PROJECT 5 – LOCALIZATION STUDIO

Create timed, reviewable multilingual media packages.

GOLDEN NOTES

1

- Buyer: educators and media teams

2

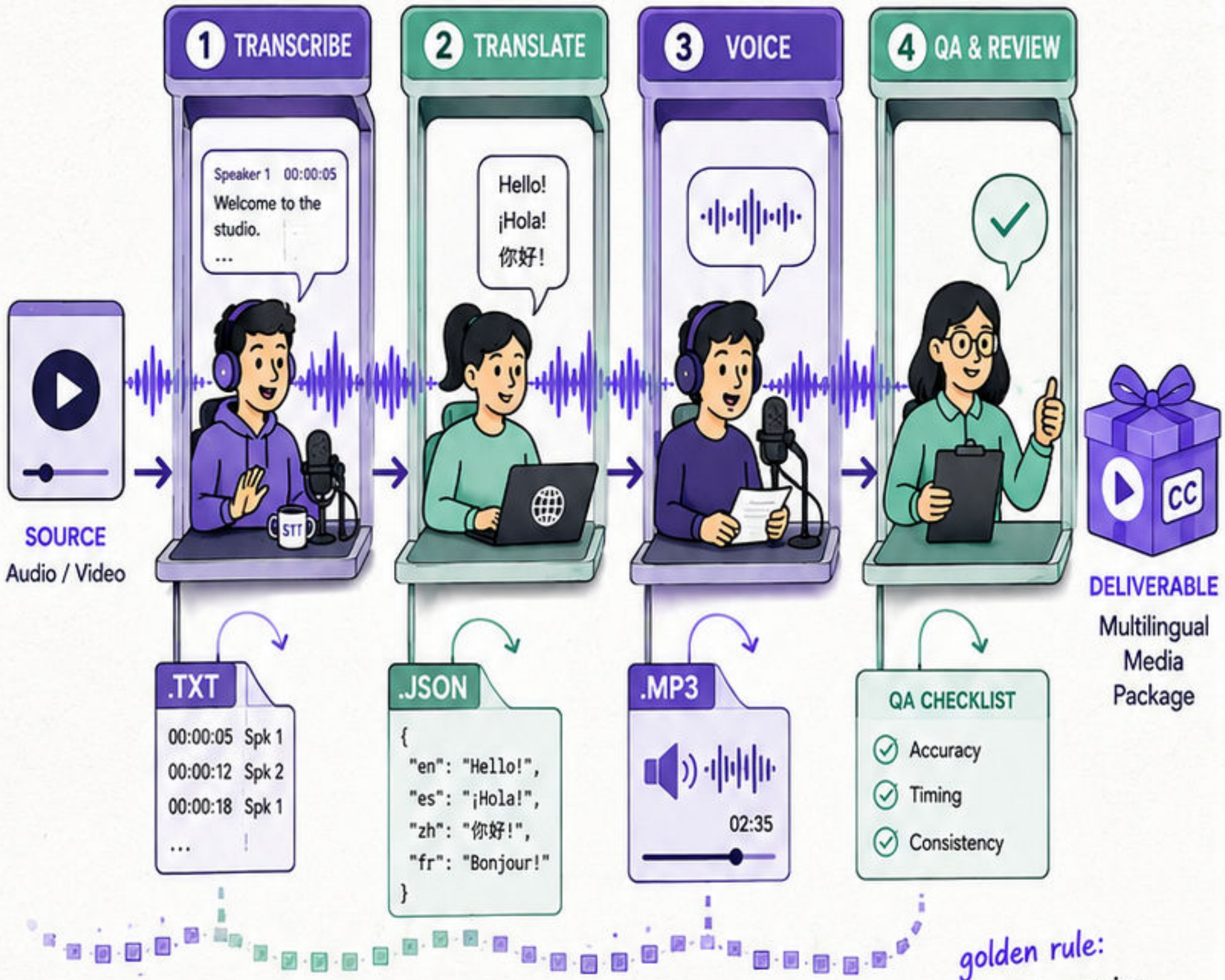
- Transcribe with speakers and timestamps

3

- Translate for meaning and audience

4

- Generate voice only with rights and consent



golden rule:
respect meaning, rights and context.

cost check

CODE

```python
def pipeline(media, locale):
    transcript = transcribe(media)
    translation = translate(transcript, locale)
    voice = synthesize_voice(translation)
    qa_report = qa_review(transcript,
                    translation, voice)
    package = bundle(transcript, translation,
                voice, qa_report)
    return package
```

media → transcript → translation → voice → QA

INPUT

Authorized audio/video + target locale

OUTPUT

Transcript, subtitle and dubbing package

COMMERCIAL LENS

Charge per source minute plus language and review tiers.

REPOSITORY

`projects/blueprints.py`

LOCALIZATION — HUMAN REVIEW MAP

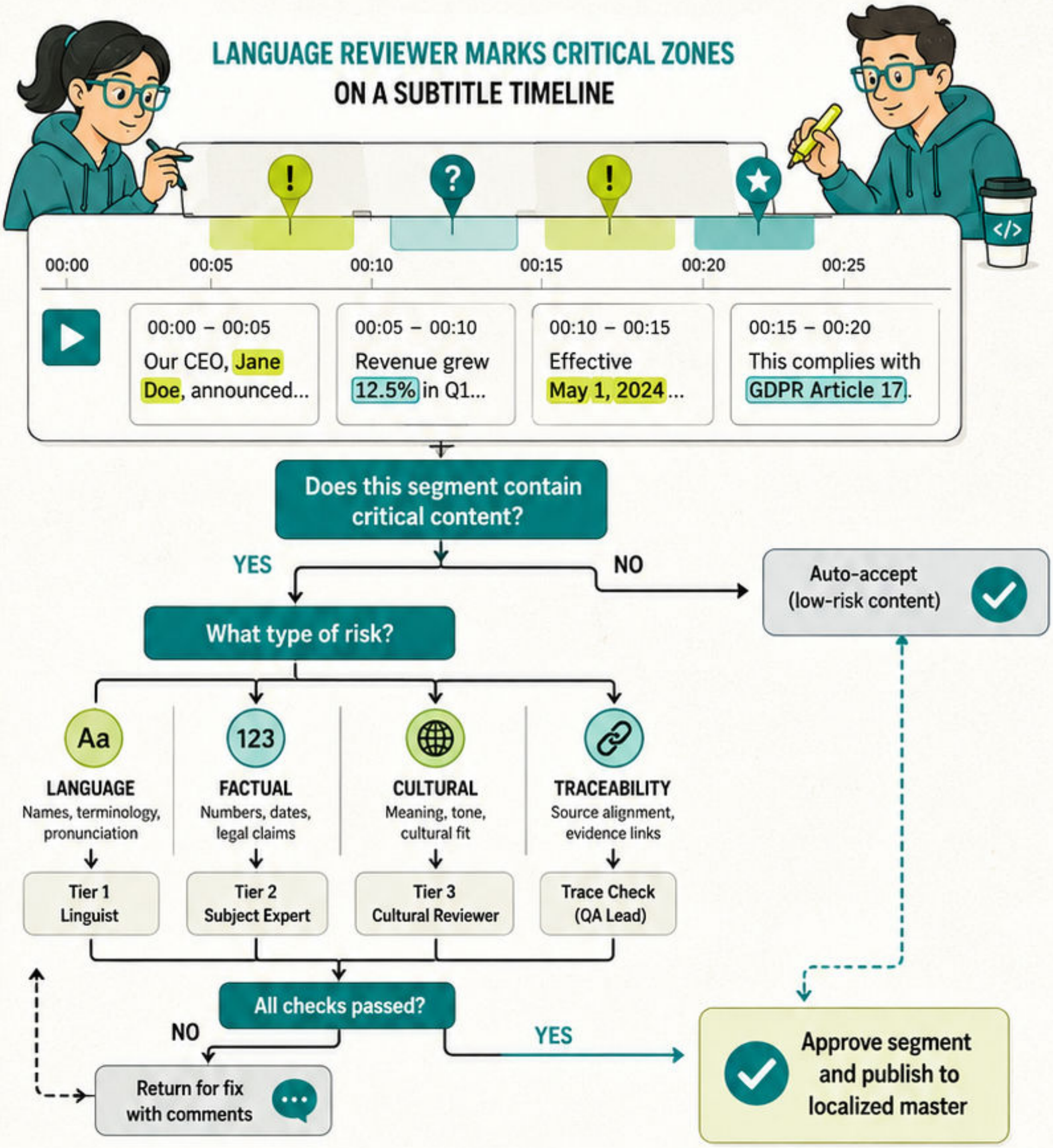
Automation accelerates the pipeline; experts protect names, claims and culture.

1 Review proper nouns and terminology

2 Verify numbers, dates and legal claims

3 Check timing, pronunciation and cultural fit

4 Keep source-to-translation traceability



CODE

segment_id links source → translation → audio

```python
class Segment(BaseModel):
    segment_id: str
    start: float
    end: float
    source_text: str
    target_text: str
    audio_uri: str
    reviewer_notes: Optional[str]

class ReviewMap(BaseModel):
    segments: List[Segment]
    status: str  # draft | in_review | approved
```

INPUT

Timed language segments

JSON

Detect critical zones

Route to right experts

Consolidate evidence

Make decision

OUTPUT

Approved localized master

COMMERCIAL LENS

Tiered review becomes a premium quality offering.

golden rule

REPOSITORY

projects/blueprints.py

PROJECT 6 – PROPERTY MEDIA STUDIO

Generate factual listing packages from approved property evidence.

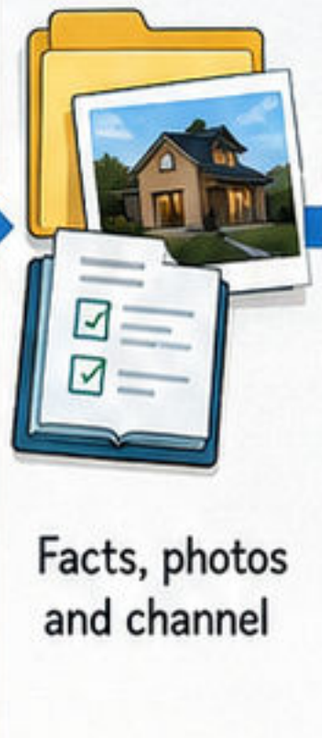
GOLDEN NOTES

- Buyer: real estate and tourism teams
- Extract only observable and supplied facts
- Create channel-specific copy and visual briefs
- Protect addresses, people, brands and disclosures

CODE

```python
def build_listing_pack(
    property_data,
    approved_media
):
  return property_data + \
  approved_media → listing_pack
```

INPUT



Facts, photos and channel

OUTPUT

Copy, image variants and short video brief

COMMERCIAL LENS

Sell per-property packages or agency subscriptions.

cost check

REPOSITORY

projects/blueprints.py

PROPERTY MEDIA — TRUST CHECKS

A beautiful listing that invents facts is a business liability.

GOLDEN NOTES

1. Lock factual fields before generation
2. Flag unsupported adjectives and claims
3. Review edited images for material changes
4. Retain approval and source provenance

TRUST DASHBOARD

FACT COVERAGE	CLAIM RISK	IMAGE INTEGRITY	SOURCE PROVENANCE
94%	6%	92%	100%

golden rule

If a fact can't be sourced, it can't be published.

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`assert every_claim in approved_source`	Generated listing candidate	Factual approved package	Trust and fast turnaround create defensible service value.	SECURITY.md

PROJECT 7 — SUPPORT TRIAGE

Classify, retrieve, draft and escalate—without silently sending risky replies.

GOLDEN NOTE 1

- Buyer: customer support teams

GOLDEN NOTE 2

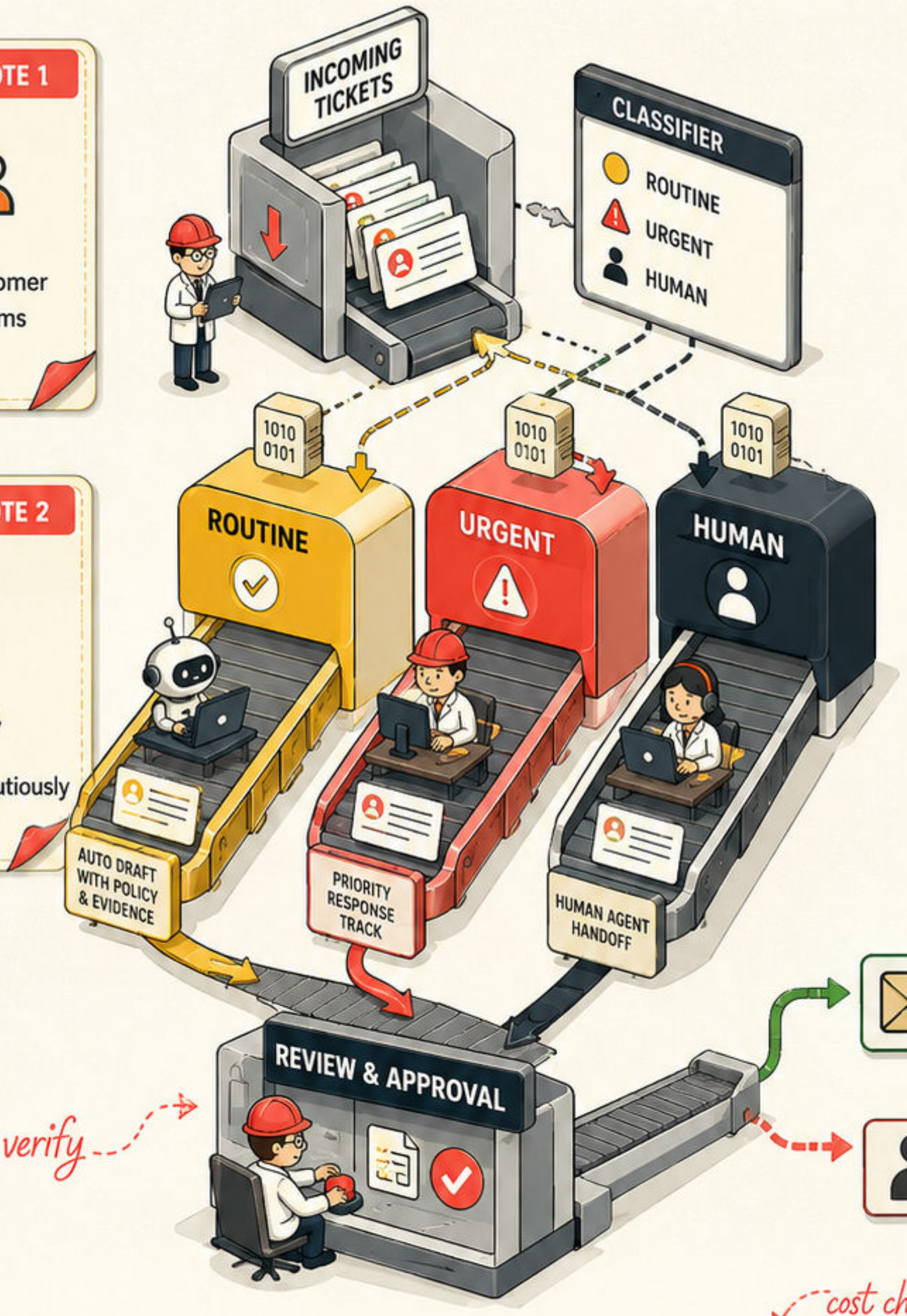
- Detect intent, urgency and sentiment cautiously

GOLDEN NOTE 3

- Retrieve approved policy and account data

GOLDEN NOTE 4

- Draft with citations; require approval for external send



cost check

CODE </>

```
1 ticket → classify →
2 retrieve → draft →
3 approve/escalate
4
5
```

INPUT

Customer message + authorized context

OUTPUT

Draft response + route + evidence

COMMERCIAL LENS

Charge per seat plus usage or resolved ticket.

REPOSITORY

projects/blueprints.py

SUPPORT TRIAGE — OPERATING METRICS

Optimize successful resolution, not automated message volume.

golden rule 1

Track first-contact resolution

golden rule 2

Measure reopen and correction rate

BEFORE: ACTIVITY FOCUSED

Activity Dashboard

Volume of messages, not outcomes.

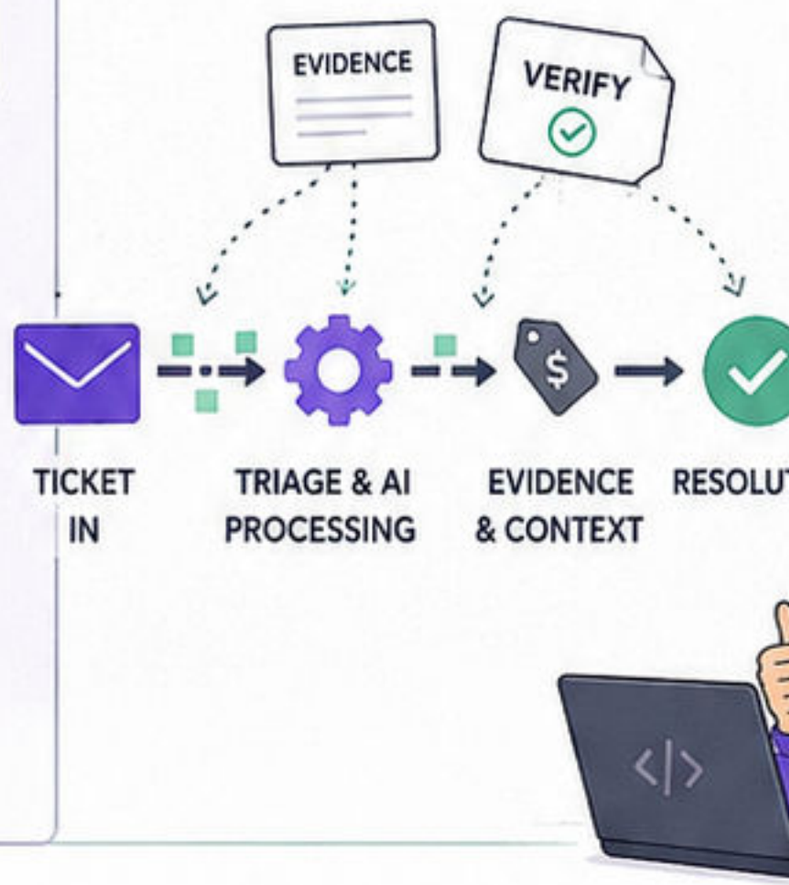
- Messages Sent 18,742
- Automations Run 6,831
- Avg. Handling Time 12m 48s
- Agent Utilization 91%

AFTER: OUTCOME FOCUSED

Outcome Dashboard

Quality outcomes and unit economics.

- First-Contact Resolution 72%
- Reopen Rate 8%
- Escalation Miss Rate 2.1%
- Cost per Verified Resolution $3.42



golden rule 3

Audit false urgency and missed escalation

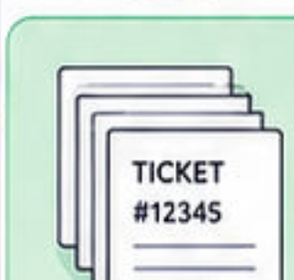
golden rule 4

Compare agent time saved with system cost

CODE

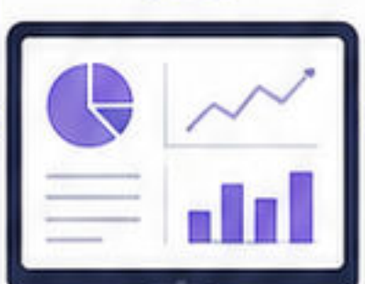
```
cost_per_resolution =
total_variable_cost /
verified_resolutions
```

INPUT



Ticket outcomes

OUTPUT



Quality and unit-economics dashboard

COMMERCIAL LENS

Outcome metrics prevent automation theater.

cost check

REPOSITORY

```
src/ai_api_playbook/
economics.py
```

PROJECT 8 — PROPOSAL & RFP COPILOT

Make every requirement traceable before drafting persuasive prose.

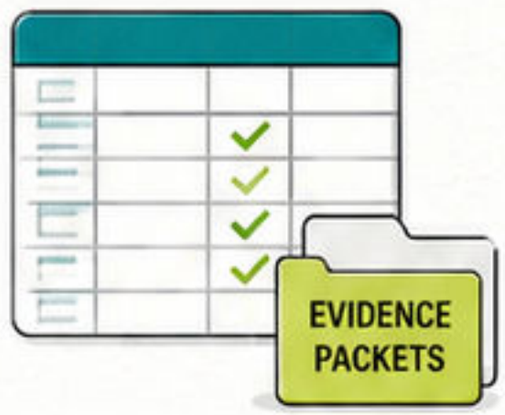
GOLDEN NOTE 1

- Buyer: agencies and B2B vendors

GOLDEN NOTE 2

- Extract requirements with page citations

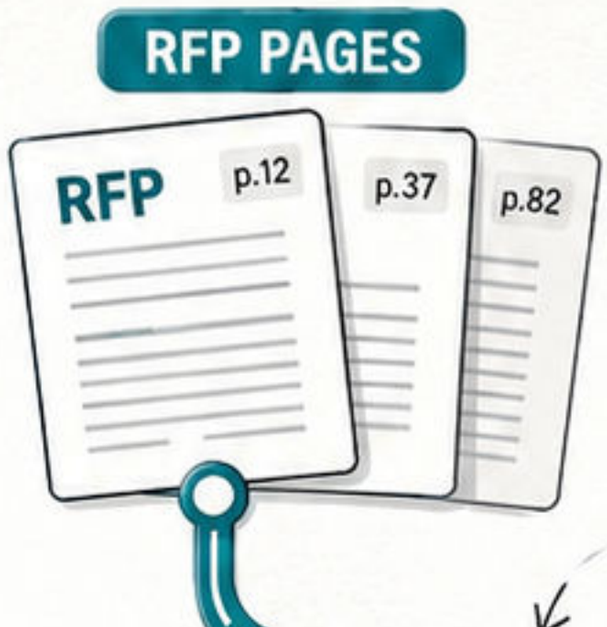
GOLDEN NOTE 3



- Build compliance and evidence matrix

GOLDEN NOTE 4

- Draft only from approved capabilities and commitments



golden rule: no requirement left behind

REQUIREMENTS EXTRACTION

REQ-023 (p.37) Data Security

REQ-047 (p.82) SLA 99.9%

COMPLIANCE & EVIDENCE MATRIX

Req ID	Requirement	Page	Commitment	Evidence	Status
REQ-023	Data Security	p.37	AES-256 encryption at rest and in transit	SEC-001.pdf	✓
REQ-047	Uptime (SLA)	p.82	99.9% monthly uptime guarantee	SLA-2024.pdf	✓

PROPOSAL STORYBOARD

verify

REVIEW & APPROVAL

cost check

CODE

```
def run():
    pipeline = [
        "RFP → requirements",
        "requirements → evidence matrix",
        "evidence matrix → draft",
        "draft → review"
    ]
    return " → ".join(pipeline)

if __name__ == "__main__":
    print(run())
```

INPUT

RFP + approved company evidence

OUTPUT

Traceable response package

COMMERCIAL LENS

Price per workspace, proposal or annual team license.

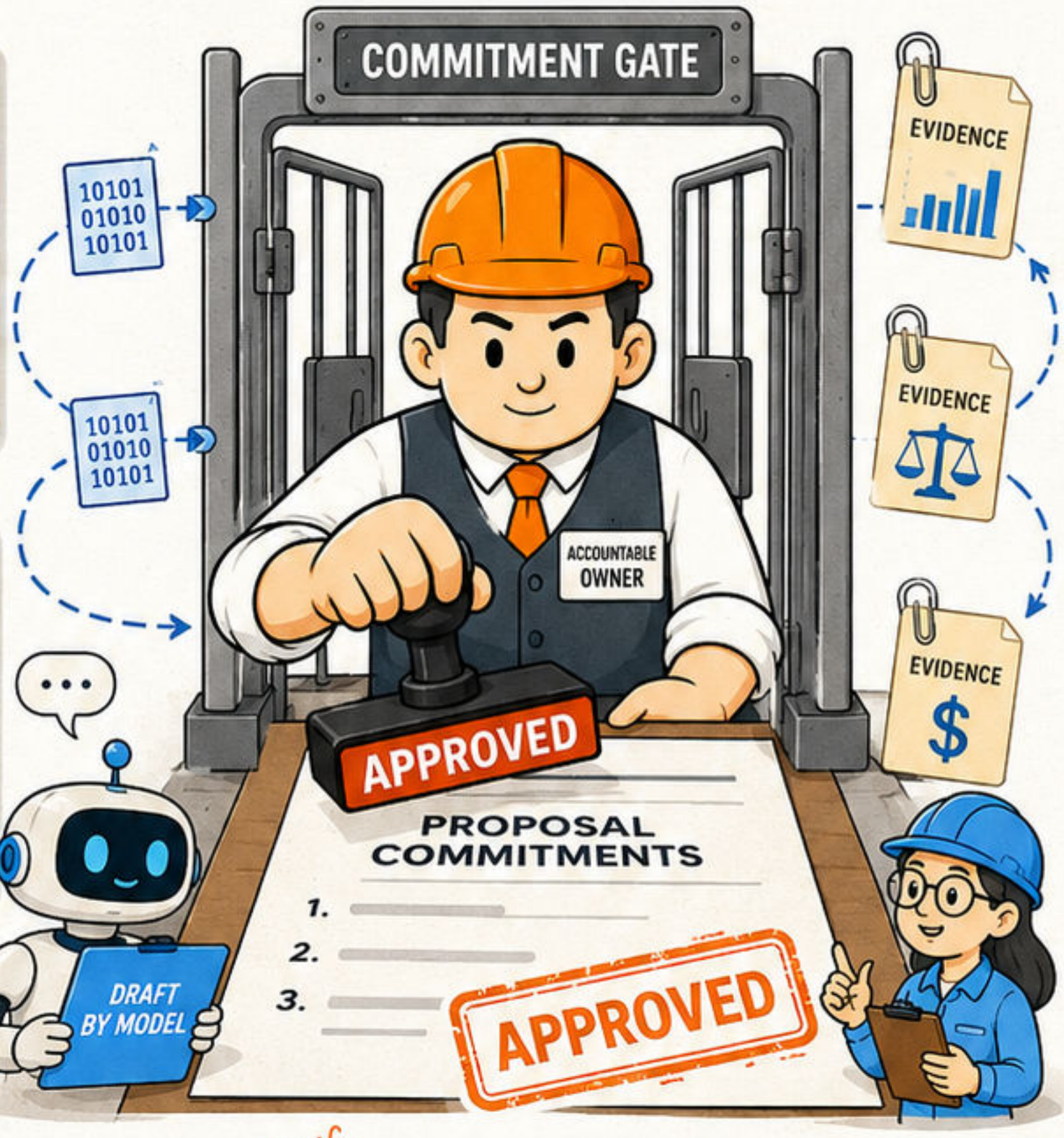
REPOSITORY

projects/blueprints.py

PROPOSAL COPILOT – COMMITMENT GATE

The model may draft; an accountable owner must approve every promise.

1. Separate facts, plans and commitments
2. Detect missing evidence
3. Flag dates, prices and legal language
4. Record approver and source for final claims



golden rule

verify

INPUT		OUTPUT	COMMERCIAL LENS
Draft commitments	`commitment.status must equal "approved"`	Approved proposal or exception list	Governance turns drafting speed into enterprise-grade value.

REPOSITORY

SECURITY.md

PROJECT 9 — NICHE AI TUTOR

Build imastery from an approved curriculum, not unlimited improvisation.

GOLDEN NOTE

- Buyer: training providers

1 CONCEPT

Clear explanations from curated, approved sources.

EVIDENCE TAG

GOLDEN NOTE

- Ground explanations in curated sources

5 FEEDBACK

FEEDBACK PACKET

Personalized feedback that turns mistakes into mastery.

2 PRACTICE

Guided practice that builds confidence step by step.

ACTION PACKET

Your learning journey

INSIGHT PACKET

EVIDENCE PACKET

4 ADAPT

Adjust difficulty and path based on observed performance.

3 ASSESS

Quizzes with answer evidence and clear rationales.

GOLDEN NOTE

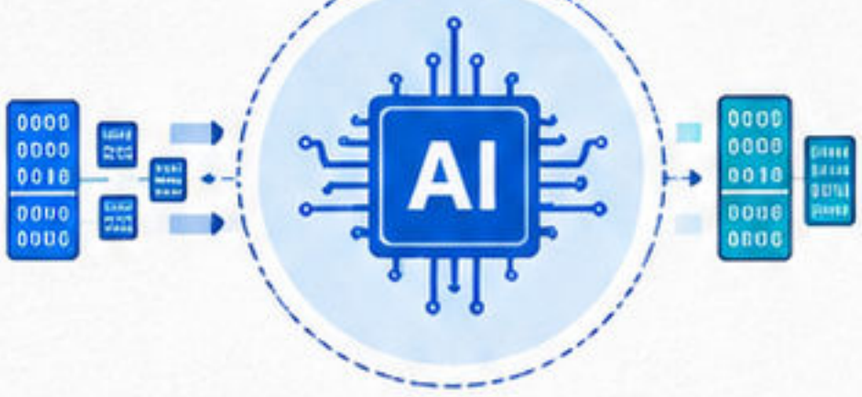
- Adapt difficulty from observed performance

GOLDEN NOTE

- Generate quizzes with answer evidence

INPUT

"Learner state + curriculum"



OUTPUT

"Lesson, practice and feedback"

CODE

```
diagnose →
teach →
practice →
assess →
adapt
```

COMMERCIAL LENS

"Charge per learner/month or license to institutions."

cost check -

REPOSITORY

"projects/blueprints.py"

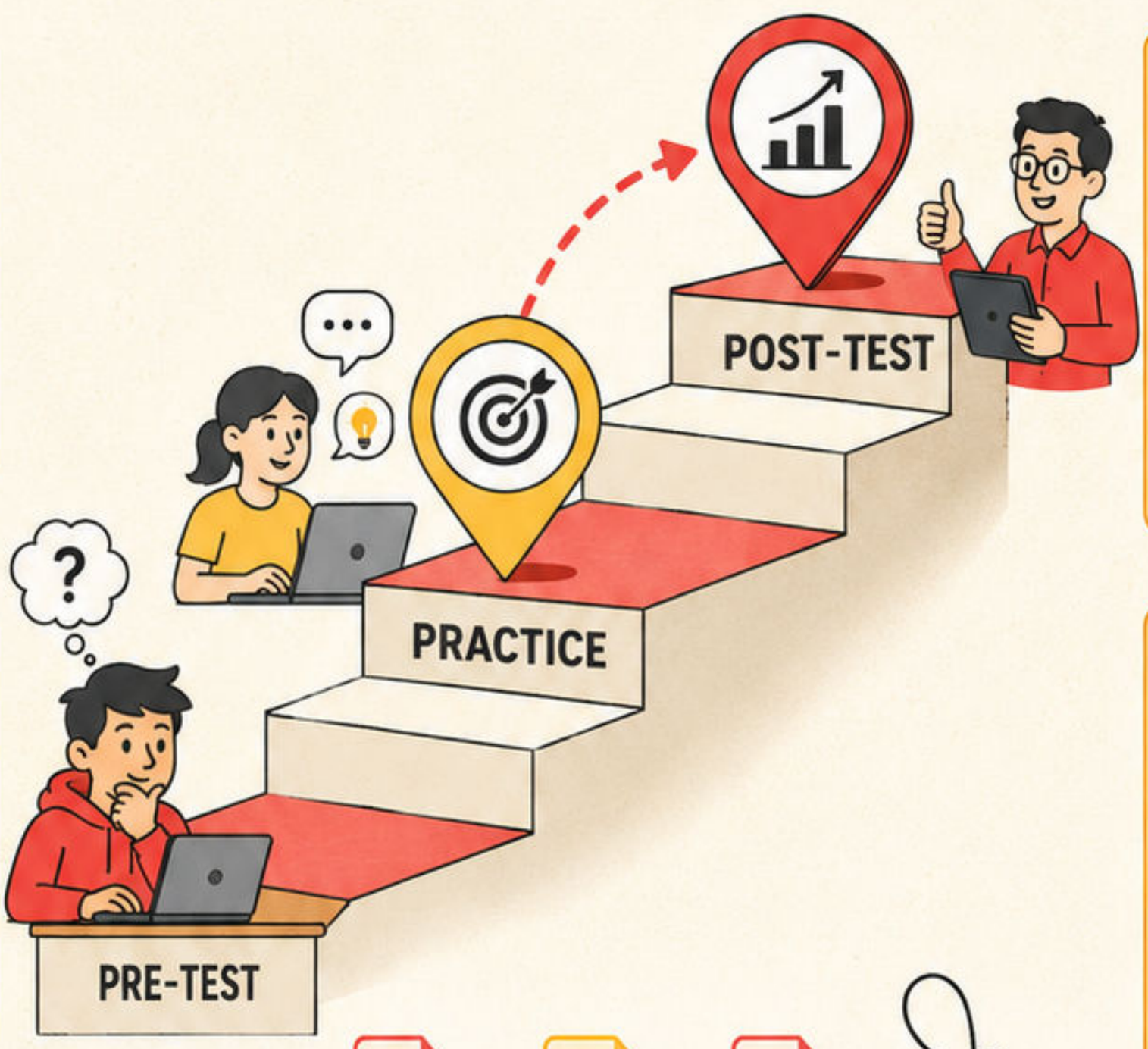
verify -

AI TUTOR — LEARNING EVIDENCE

Engagement is not the same as learning.

- Measure pre/post performance

- Use delayed retrieval practice



- Track misconception categories

- Let instructors inspect sources and feedback

CODE

```
learning_gain =
post_score - pre_score
```

INPUT

"Assessment events"

OUTPUT

"Evidence of mastery and next step"

COMMERCIAL LENS

Measured learning outcomes support durable educational products.

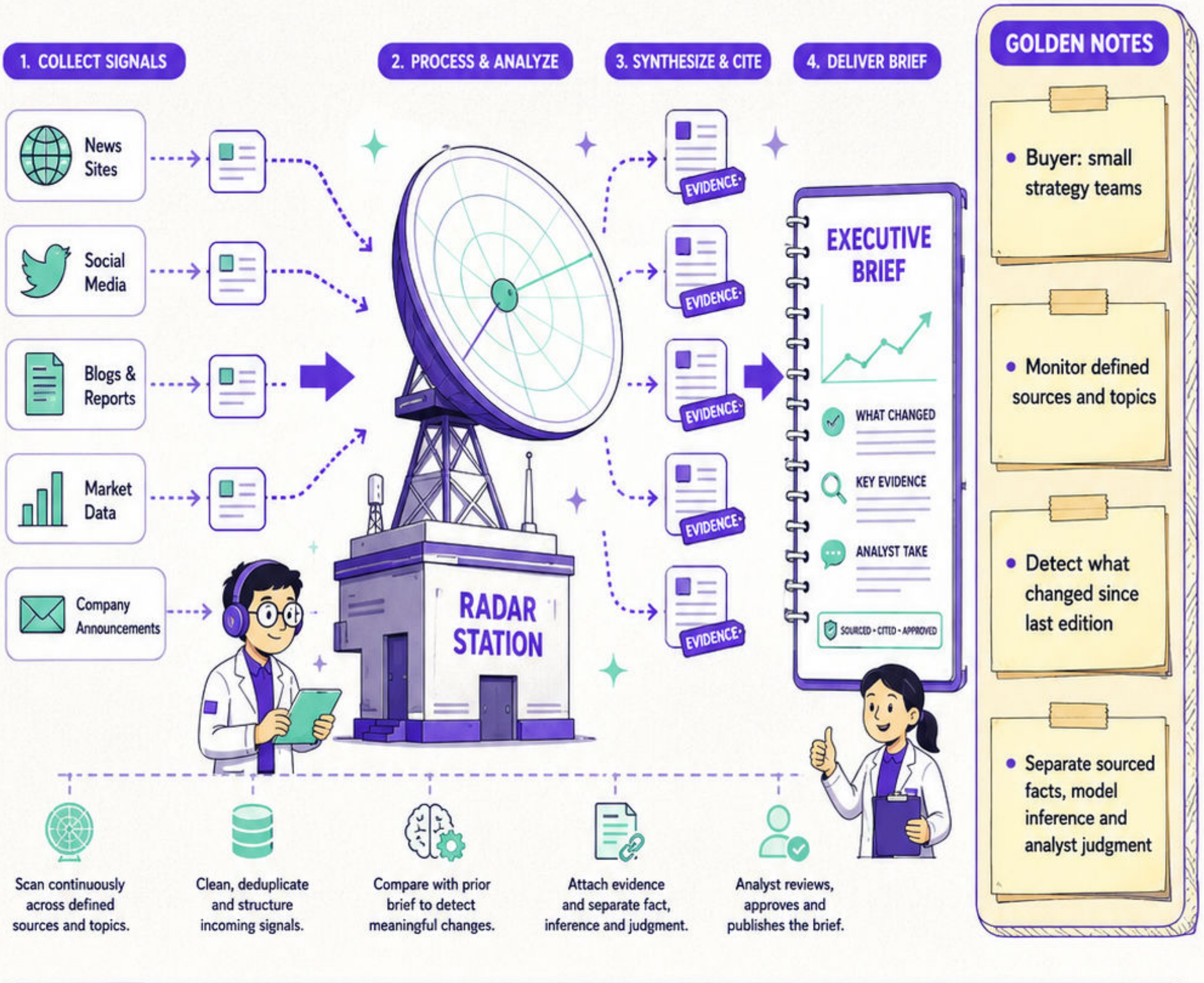
REPOSITORY

`src/ai_api_playbook/evaluation.py`

PROJECT 10 — MARKET INTELLIGENCE BRIEF

Convert scattered signals into a cited recurring decision product.

1. COLLECT SIGNALS
2. PROCESS & ANALYZE
3. SYNTHESIZE & CITE
4. DELIVER BRIEF



Scan continuously across defined sources and topics.

Clean, deduplicate and structure incoming signals.

Compare with prior brief to detect meaningful changes.

Attach evidence and separate fact, inference and judgment.

Analyst reviews, approves and publishes the brief.

GOLDEN NOTES

- Buyer: small strategy teams
- Monitor defined sources and topics
- Detect what changed since last edition
- Separate sourced facts, model inference and analyst judgment

CODE

```
collect →
deduplicate →
compare →
synthesize →
cite →
approve
```

INPUT

"Current sources + prior brief"

OUTPUT

"Cited change report"

COMMERCIAL LENS

Sell weekly or monthly subscriptions by niche.

REPOSITORY

"projects/blueprints.py"

MARKET INTELLIGENCE — TRUST & RETENTION

Customers renew when the brief is timely, relevant and auditable.

1

- Track source freshness

2

- Score claim-to-source support

3

- Collect which insights changed decisions

4

- Archive editions and corrections transparently

RELEVANCE
Right topics, right audience

TRUST
Auditable evidence, transparent updates

ACTIONABILITY
Clear takeaways that inform decisions

SUBSCRIPTION FLYWHEEL
Relevance drives value. Evidence builds trust. Actionability drives renewal.

EVIDENCE

EVIDENCE

Data in

Decisions out

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`renewal_value = relevance × trust × actionability`	"Brief usage and feedback"	"Improved source and topic policy"	"A narrow trusted niche beats a generic flood of summaries."	`projects/blueprints.py`

CAPSTONE — AI PUBLISHING ENGINE

Turn an authorized manuscript into a governed publishing package.

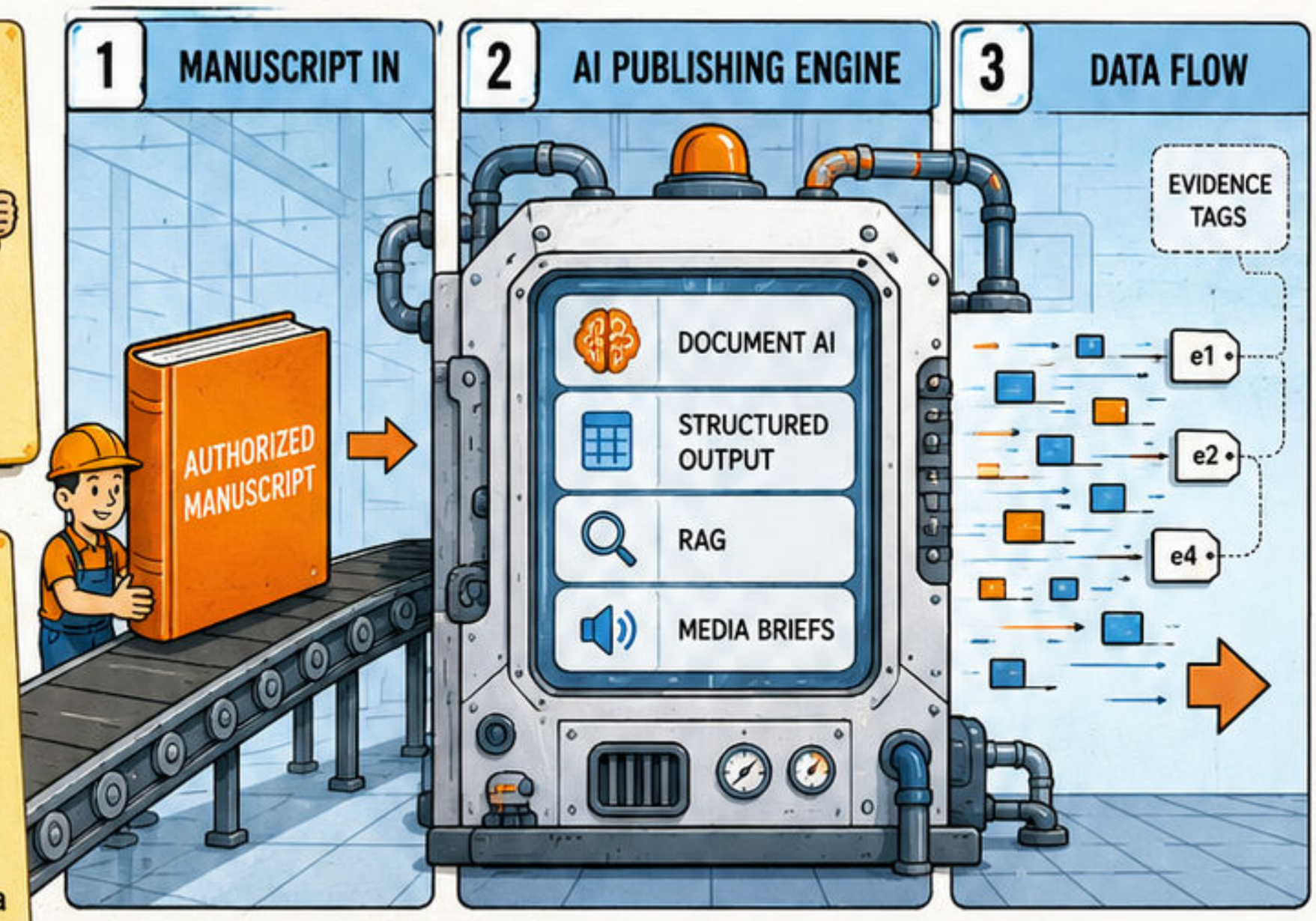
- User: author or publisher

- Input: owned manuscript and metadata

1 MANUSCRIPT IN

2 AI PUBLISHING ENGINE

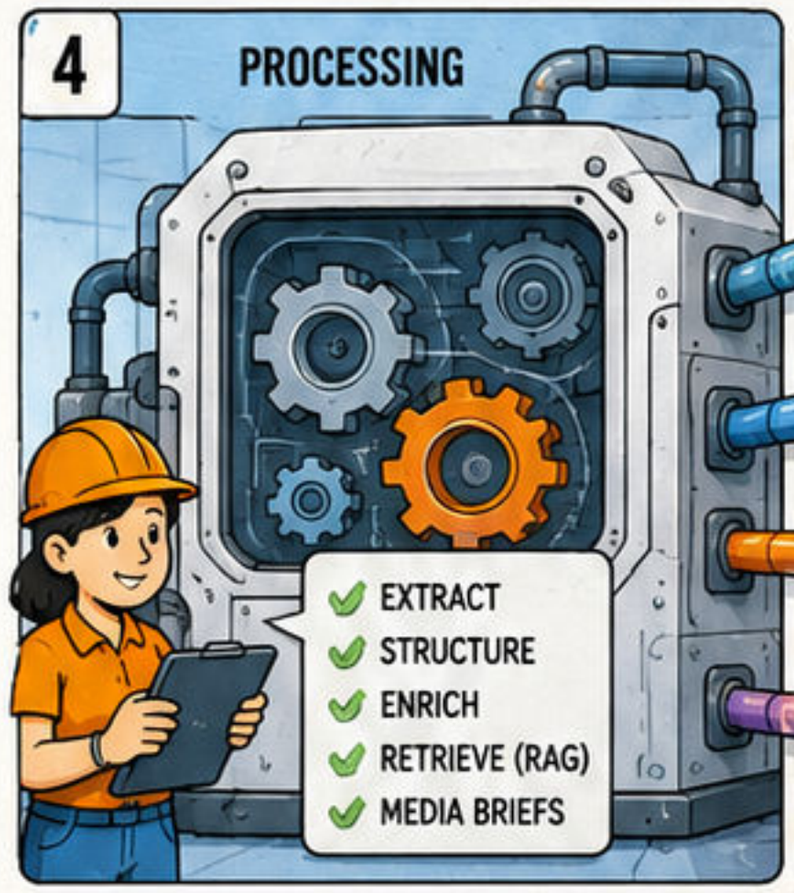
3 DATA FLOW



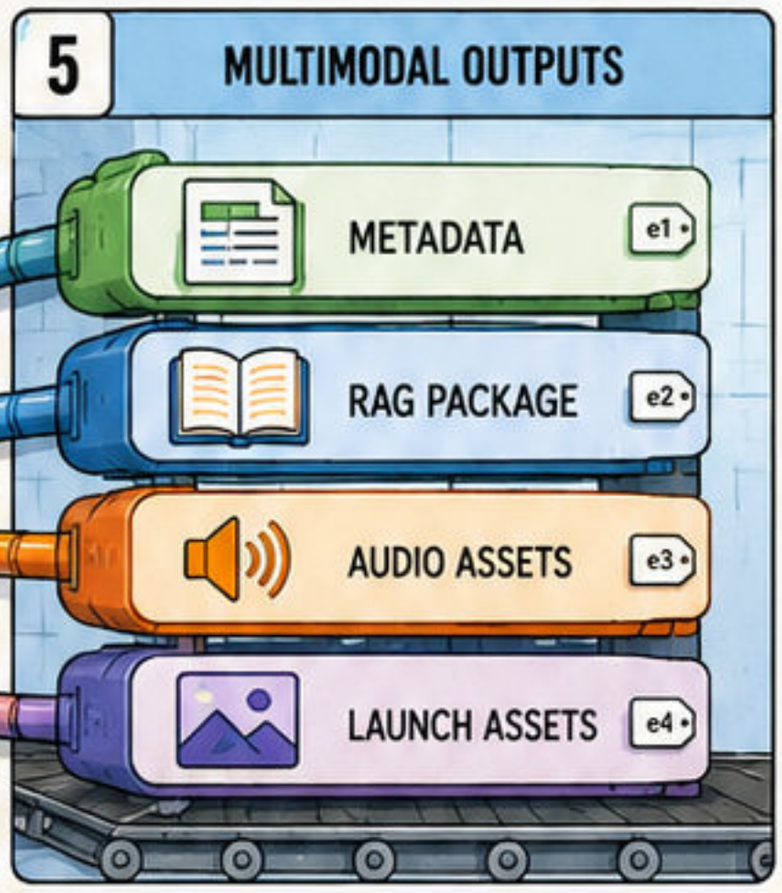
- Core: document AI, structured output, RAG and media briefs

- Output: human-review package—never automatic publication

4 PROCESSING



5 MULTIMODAL OUTPUTS



6 HUMAN REVIEW GATE

CODE

```
manuscript →
extract →
structure →
enrich →
review →
export
```

INPUT

"Authorized book/PDF"

OUTPUT

"Publishing and knowledge-commerce package"

COMMERCIAL LENS

"Offer as author service, publisher tool or SaaS."

REPOSITORY

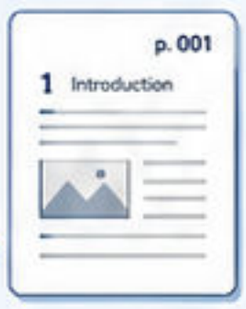
projects/publishing_engine/ README.md

CAPSTONE — DOCUMENT TO KNOWLEDGE

Preserve provenance while turning pages into searchable evidence.

- Fingerprint the source version
- Extract headings, tables and page locations
- Chunk with section and page metadata
- Build retrieval tests from real reader questions

MANUSCRIPT PAGES



TRACEABLE CHUNKS

1.1 What is RAG?
p. 001
sec: 1.1 id: c001
v1.0

1.2 Why Provenance Matters
p. 004
sec: 1.2 id: c002
v1.0

3.4 Indexing Strategy
p. 042
sec: 3.4 id: c015
v1.0

7.2 Results Overview
p. 128
sec: 7.2 id: c067
v1.0

A.1 Prompts Library
p. 256
sec: A.1 id: c201
v1.0

SEARCH INDEX

- Semantic Embeddings
- Metadata & Provenance
- Fast Retrieval & Ranking

SOURCE (PAGES) → EXTRACT & PARSE → CHUNK & TAG → INDEX & STORE → RETRIEVE & ANSWER

CODE

```
source_hash +
page +
section +
chunk_id
```

INPUT

"Manuscript pages"

OUTPUT

"Versioned RAG manifest"

COMMERCIAL LENS

A reliable book assistant extends the commercial life of content.

REPOSITORY

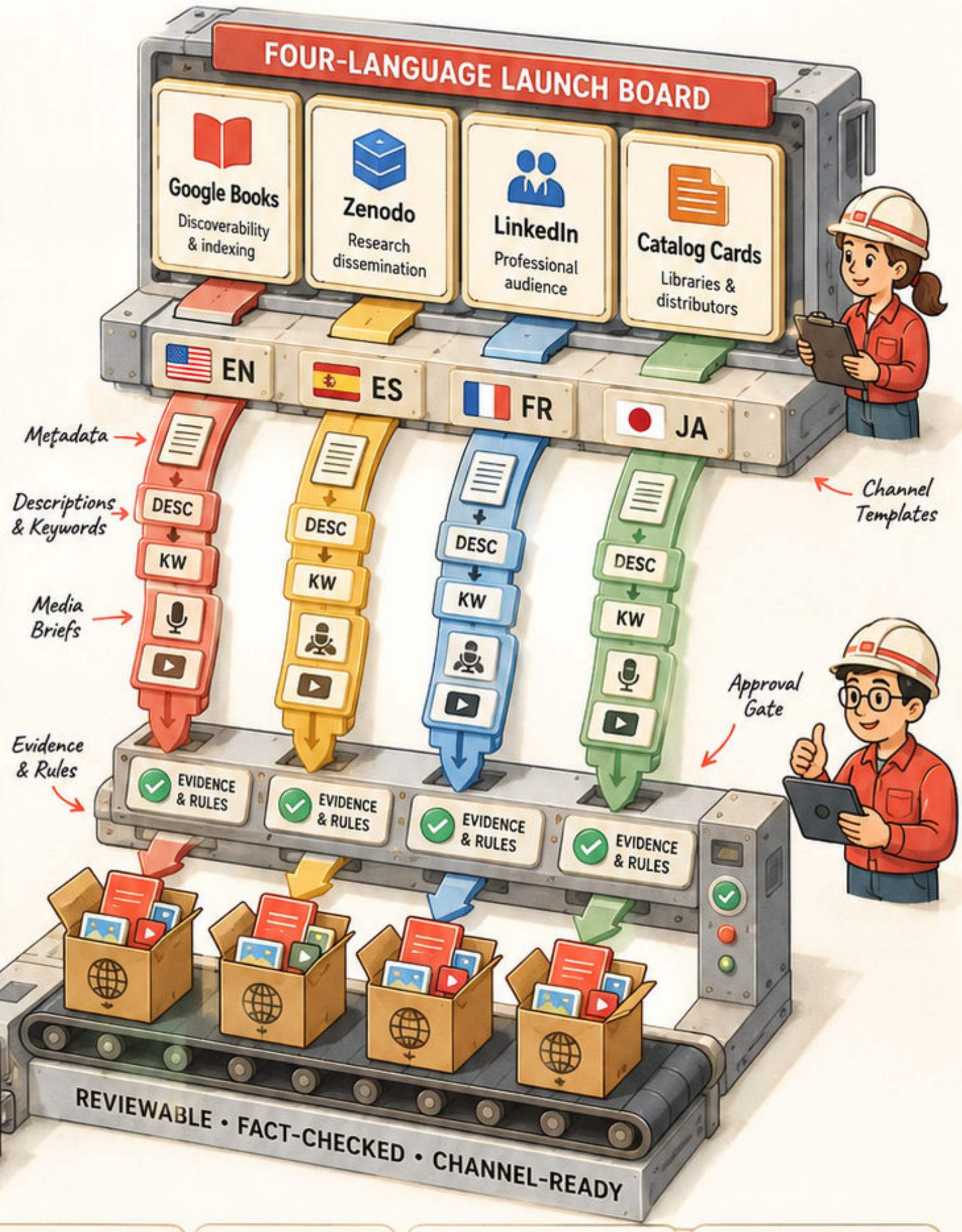
</>

projects/
publishing_engine/
pipeline.py

CAPSTONE — MULTILINGUAL COMMERCE PACK

Generate channel-specific assets without losing factual control.

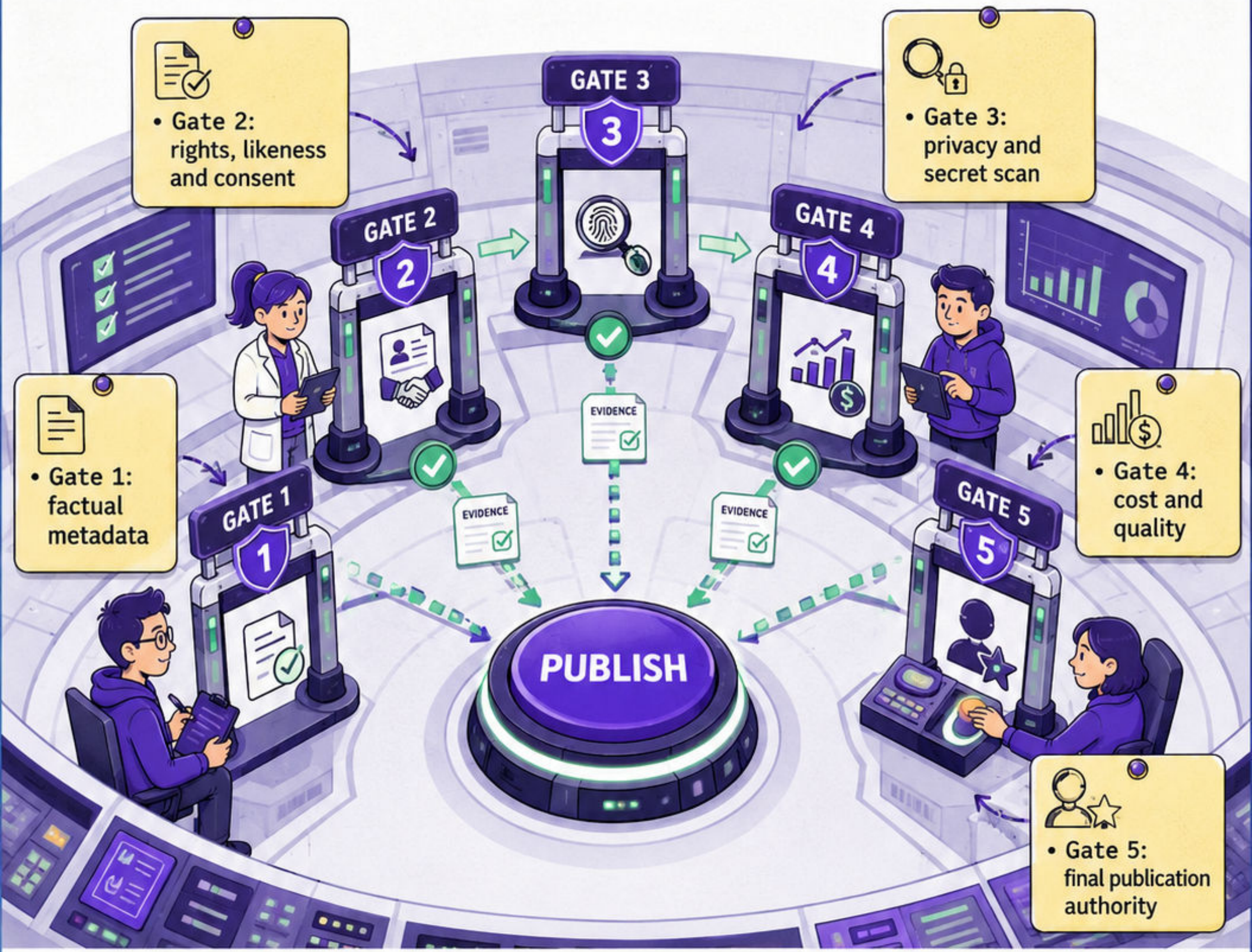
- Create schema-valid book metadata
- Generate language-specific descriptions and keywords
- Prepare audio, image and video briefs
- Review facts, rights, locale and platform rules



CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`metadata → locale packs → channel templates → approval`	"Approved manuscript metadata"	"Reviewable multilingual launch assets"	"Sell premium packaging by language and channel."	`projects/ publishing_engine/ pipeline.py`

CAPSTONE — HUMAN APPROVAL SYSTEM

The final product is not the generation; it is the approved state transition.



CODE

```
if all(gates):
    export();
else:
    review_queue()
```

INPUT

"Generated package + evidence"

OUTPUT

"Approved export or explicit exceptions"

COMMERCIAL LENS

"Human approval preserves trust while automation creates speed."

REPOSITORY

projects/publishing_engine/pipeline.py

RELEASE, CITATION & DOI

Freeze a verified version so others can run, inspect and cite the same work.

1. Test and document before tagging
2. Use semantic versioning and a clear changelog
3. Include CITATION.cff, license and ORCID
4. Create separate linked records for software and book

`git tag v1.0.0`

VERIFIED
Tests passed
Docs complete

GitHub Release → RELEASE v1.0.0

Zenodo Archive

DOI
10.5281/zenodo.1234567
CITE THIS WORK

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY
`git tag v1.0.0` → GitHub Release → Zenodo DOI	"Verified source tree"	"Citable archived software release"	A DOI strengthens discoverability and scholarly attribution.	docs/zenodo-release.md

FROM API CALL TO DURABLE VALUE

The future belongs to systems that combine intelligence with evidence, control and economics.

- Begin with one measurable customer pain

- Build a safe, observable workflow

- Evaluate quality and cost per successful job

- Publish evidence, improve continuously and earn trust

CODE	INPUT	OUTPUT	COMMERCIAL LENS	REPOSITORY

`capability`
`+ workflow`
`+ evidence`
`+ governance`
`+ distribution`
`= value`
	"A useful idea and disciplined engineering"	"A trusted product—not merely a demo"	"The API is rented intelligence; the system around it is your lasting asset."	README.md

From API Calls to Trusted Products

A VISUAL ENGINEERING GUIDE TO BUILDING REAL AI PRODUCTS

AI is no longer the future—it's the present. But moving from a single API call to a reliable, secure, and scalable product is where most projects fail. This book is your visual engineering guide to bridging that gap.

Through 100 carefully crafted visual lessons, you'll explore **multimodal AI, agents, RAG, evaluation, governance, security, architecture,** and **cost awareness**—everything you need to design, build, and ship AI-powered products that users trust and businesses value.

Whether you're an engineer, product builder, or technical leader, this book equips you with practical patterns, real-world examples, and production-ready insights to turn ideas into robust systems and commercially useful products.

Free source code for this book is available at:
https://faramarzkowsari.github.io/ai-api-playbook/

INSIDE THE BOOK

- 100 visual lessons
- multimodal workflows
- agents and RAG
- evaluation and governance
- real-world product thinking

ABOUT THE AUTHOR

Faramarz Kowsari is an author, Software Engineer and AI researcher based in Istanbul. Focusing on the intersection of technology, education, and personal growth, he has published over 80 digital titles on international platforms. His areas of expertise span Artificial Intelligence, prompt engineering, modern trading strategies (Smart Money Concepts & algorithmic trading), as well as classical literature and mindfulness. In addition to writing, he develops web-based educational tools and creates specialized instructional video content.

Official Profiles & Repositories:

- Wikidata: https://www.wikidata.org/wiki/Q140389378
- ORCID: https://orcid.org/0000-0003-1692-0453
- Google Scholar: https://scholar.google.com/citations?user=G7tP5WMAAAJ&hl=en
- GitHub: https://github.com/FaramarzKowsari
- LinkedIn: https://www.linkedin.com/in/faramarzkowsari
- Google Books: https://play.google.com/store/search?q=Faramarz_Kowsari&c=books

Google Books Key • GGKEY: DTJJHJUDW0Y

DOI: 10.5281/zenodo.21431944