— THE METHOD

The Responsible AI Readiness Framework

Responsible AI isn't a policy document — it's safety, governance and accountability delivered as production-grade code. This is how it gets measured: **six dimensions, scored 0–100**, where every check is rated on whether it is absent, ad-hoc, managed, or actually enforced in code. The output is a number you can act on and a list you can work through.

— HOW IT WORKS

Rated on enforcement, not on intention.

Each dimension has a short checklist. Every item is rated on a four-level maturity scale, the dimension score is the percentage of points achieved, and the overall Readiness Score is the weighted average. The question behind every rating is the same: is it enforced by the system, or is it a promise a person has to keep?

Maturity scale

Applied to every individual check.

0	**Absent**	Not done.
1	**Ad-hoc**	Done once, manually, undocumented, not repeatable.
2	**Managed**	Repeatable and documented, but not enforced automatically.
3	**Production-grade**	Enforced in code / CI, observable, and reproducible.

Readiness bands

What the total score means in practice.

Demo · 0–40	Impressive in a meeting; not safe to ship.
Piloting · 41–70	Works, but not defensible to a regulator.
Production-ready · 71–90	Safe to run with real users and data.
Audit-ready · 91–100	Defensible under the EU AI Act.

Weights, configurable per engagement — Guardrails 20% · Governance 20% · Data 15% · Observability 15% · Testing 15% · Reliability 15%

— THE SIX DIMENSIONS

What gets checked.

01 Data quality & lineage 15%

An AI decision is only as defensible as the data behind it.

- ☐ Schema contracts on every ingestion boundary (types, ranges, required fields), enforced — not assumed.
- ☐ Input validation rejects or quarantines bad records instead of silently passing them downstream.
- ☐ End-to-end lineage: any model output can be traced back to the source data and transformations that produced it.
- ☐ Train/serve feature parity is enforced and proven by test — no training-serving skew.
- ☐ Sensitive fields are classified and handled at the data layer, not just at the model.

DEMO LOOKS LIKE

A notebook reading a CSV with no validation.

PRODUCTION-GRADE LOOKS LIKE

Typed contracts plus lineage, so every prediction is explainable from data to output.

02 Guardrails & safety 20%

The system must refuse to produce unsafe or ungrounded output.

- [ ] Input guardrails: prompt-injection, jailbreak, and out-of-scope request handling.
- [ ] Output guardrails: PII redaction, toxicity and relevance checks, and response grounding against an approved source.
- [ ] An evaluation harness scores outputs against a labelled set before any release — not just vibes.
- [ ] A human-in-the-loop or escalation path exists for low-confidence and high-impact decisions.
- [ ] LLM judgement is scoped to where it adds value over deterministic logic — and bounded everywhere else.

DEMO LOOKS LIKE

Raw model output returned straight to the user.

PRODUCTION-GRADE LOOKS LIKE

Guardrails enforcing redaction and grounding, with an eval gate in CI that fails the build.

03 Observability & drift 15%

You cannot be accountable for a system you cannot see.

- ☐ Structured logging of inputs, outputs and decisions — with PII handled — for every inference.
- ☐ Dashboards for latency, error rate, cost and volume: live, not on request.
- ☐ Data drift and model/output-quality drift are monitored, with thresholds.
- ☐ Alerting fires on degradation before a user reports it, and routes to a named owner.
- ☐ Each decision is traceable end to end: request → features → model → guardrails → output.

DEMO LOOKS LIKE

`print()` statements and a hope.

PRODUCTION-GRADE LOOKS LIKE

Dashboards, drift detection and alerting wired from day one.

04 Governance as code 20%

Accountability must be provable on demand, not promised in a PDF.

- ☐ Access control and data grants defined declaratively and version-controlled.
- ☐ Audit trail: who, what and when for data access, model changes and deployments — queryable.
- ☐ EU AI Act technical documentation generated from the system and kept in sync with the code.

- [ ] Model and dataset cards: intended use, limitations, known risks, evaluation results — versioned.
- [ ] A model-promotion gate enforces quality and safety thresholds before anything reaches production.

DEMO LOOKS LIKE

A shared admin login and a slide about "compliance later".

PRODUCTION-GRADE LOOKS LIKE

Governance as code, and an audit trail you can hand to a regulator.

05 Tested & reproducible 15%

Behaviour you can't reproduce, you can't trust.

- [ ] Core logic separated from cloud and framework execution, covered by a credential-free local test suite.
- [ ] Tests are CI-gated: nothing merges or ships on red.
- [ ] All infrastructure is code — no console-click deployments.
- [ ] Versions are pinned and builds are reproducible across machines.
- [ ] Behaviour-preserving refactors are verified by test, not assumed.

DEMO LOOKS LIKE

"It worked on my machine last Tuesday."

PRODUCTION-GRADE LOOKS LIKE

CI-gated suites and full Infrastructure as Code, reproducible on demand.

06 Self-healing reliability 15%

A trustworthy system recovers without heroics.

- [ ] Idempotency: re-running a job produces the same result, not duplicates or corruption.
- [ ] Checkpointing and exactly-once semantics where state matters.
- [ ] Automated remediation for known failure classes: retry, backoff, self-repair.
- [ ] Isolated state and minimised blast radius, so failures don't cascade.
- [ ] Guarded destructive operations — typed confirmation or approval — to prevent accidental teardown.

DEMO LOOKS LIKE

A manual rerun and a 3am page.

PRODUCTION-GRADE LOOKS LIKE

Idempotent, checkpointed pipelines with automated recovery.

— APPLIED

How my own builds score against it.

The framework isn't theory I wrote for clients and exempted myself from. Six public repositories, roughly 1,400 credential-free CI-gated tests, each one demonstrating specific dimensions — code and CI open to inspection.

BUILD	STRONGEST DIMENSIONS IT DEMONSTRATES
FintelliGuard	**Guardrails & safety** — 19/19 adversarial probes blocked, 0/6 benign controls wrongly blocked · **Governance** — Annex IV and Art. 12 records generated from code, promotion gate · **Observability** — drift · **Testing** — 592 CI tests
Self-Healing Multi-Cloud Agents	**Self-healing reliability** — bounded, fail-closed heal loop with an evidence gate · **Testing** — offline eval harness over 17 failure classes with no LLM, cloud or keys; 380 tests
Multi-Cloud Governance Platform	**Governance as code** — a PII/least-privilege analyzer as a merge gate, cross-checked in OPA/Rego, expiring signed exceptions, packaged as a CLI and GitHub Action · **Guardrails** — a bounded copilot · **Testing** — 137 CI tests
Fleet Risk Lakehouse	**Data quality & lineage** — quarantine, SCD-2, PSI drift · **Governance** — explainable risk index, GDPR Art. 9 column masks · **Reliability** — exactly-once · **Testing** — 165 tests
Real-Time Telemetry Pipeline	**Data quality & lineage** — declared contract, dead-letter with the reason · **Observability & drift** — z-test detector, Grafana, Slack · **Tested & reproducible** — keyless, 100% IaC; 102 tests
Contract-Driven Data Pipeline	**Data quality & lineage** — contract, quarantine carrying the violated rule, generated PII dictionary · **Testing** — 47 tests