BreakPoint Strategy Recap (Extended)

1. Core Identity

- BreakPoint is a local decision engine that tells developers whether an AI change is safe to ship.
- Short-term positioning: Dev infrastructure (pytest for AI changes).
- Long-term potential: Policy enforcement and CI deployment gatekeeper.

2. Mode Architecture (Single Library, Two Exposure Layers)

- One Python library, not two separate packages.
- Lite Mode (default): Opinionated, zero-config, frictionless entry.
- Full Mode (advanced): Configurable, policy-driven, CI-oriented.
- Full mode activated via flags or config, not separate installation.

3. Lite Mode Philosophy

- Designed for indie developers.
- No configuration required.
- Policies included: Cost delta, PII detection, Basic drift detection.
- Clear ALLOW / WARN / BLOCK output.
- Override allowed but explicit and deliberate.
- No waivers, presets, or persistent policy edits.

4. Full Mode Philosophy

- Designed for power users and CI workflows.
- Supports presets, environments, waivers, strict mode.
- Allows contract enforcement and latency policies.
- Enables persistent configuration and governance patterns.

5. Override Strategy

- Overrides allowed in Lite but must be explicit (e.g., --accept-risk cost).
- User must name the risk explicitly.
- Override should feel deliberate, not silent.
- Full mode may support persistent waivers and reasoning logs.

6. Product Personality

- Voice of a cautious senior engineer.
- Calm, specific, deterministic.
- Never authoritarian or overly dramatic.
- Transparent explanations for all decisions.

7. Risk Philosophy

- BLOCK should be rare and clearly justified.
- WARN should highlight meaningful but non-critical changes.
- Avoid false negatives more than avoid false positives.
- Flag surprising or invisible risks, not trivial noise.

8. Learning Strategy (Early Stage)

- Optimize for adoption and observable usage.
- Gather signal on override frequency and policy triggers.
- Do not implement adaptive intelligence yet.
- Future Plus layer may analyze override patterns and suggest adjustments.

9. Monetization Strategy (Local-First)

- Free Lite version provides essential safety checks.
- Pro/Plus unlocks deeper intelligence and workflow integration.

- Revenue must justify prevention of cost explosion and developer embarrassment.
- Advanced semantic drift detection (embedding-based).
- Stronger PII detection (beyond regex).
- Strict CI enforcement mode.
- Professional reporting and structured contract enforcement.

10. Value Proposition for Paid Tier

- Prevents hidden cost explosions (token spikes, model misuse).
- Prevents embarrassing production failures (format drift, unsafe outputs).
- Acts as a professional guardrail rather than a basic safety net.
- Provides higher confidence in shipping AI changes.

11. Long-Term Evolution Path

- Stage 1: Local decision tool.
- Stage 2: CI helper.
- Stage 3: Policy enforcement layer.
- Stage 4: Organization-wide governance.

12. Strategic Recommendation

- Ship one library.
- Default to Lite mode.
- Keep Full mode available but not over-marketed.
- Focus on clarity, determinism, and trust over feature expansion.
- Monetize depth of intelligence, not local execution.