README — Step 3: Risk Scoring → Strategy Selection

This README explains how to implement **Step 3** in the AI-driven CI/CD pipeline: converting build and artifact data into a `risk_score` and selecting a deployment strategy (`rolling | canary | blue-green`).

🎯 Goal

- Convert **build-info / artifact metadata** into a **risk_score (0–100)**.
- Select the **deployment strategy** + **pace schedule**.
- Ensure the decision is **auditable, versioned, and testable** in shadow mode before enforcement.

Owners

- **Platform/CI** → pipeline implementation
- **SRE/Release** → policy + runbooks
- **Service owners** → SLOs, criticality labels
- **Security** → vulnerability gating

Deliverables

- `risk-inputs.json` (from Step 2)
- `risk-decision.json` (output of this step)
- Policy files (versioned) + policy registry entry
- Evidence attached to artifact in Artifactory

🅰 Gather & Normalize Inputs

1. Collect `build-info.json` / `risk-inputs.json` from Artifactory.
2. Ensure it contains:
3. Commit SHA, buildId, repo, branch, author
4. Per-platform artifacts + checksums
5. Unit/integration test results (failed, flaky)
6. Coverage delta vs baseline
7. SBOM & dependency changes
8. Static/security scan summary
9. Files changed + LOC changed

10. Historical failure rate
11. Environment targets (arm/linux/windows)
12. Hardware/emulator test results (if available)
13. Add runtime/context signals:
14. Current production traffic/QPS
15. Calendar context (peak vs maintenance window)
16. Service criticality tag
17. Last rollback flag

📄Deliverable: `risk-inputs-for-decision.json`

🅱 Decide Scoring Approach

- **Phase 1 (Heuristic):** weighted formula → immediate rollout; good for shadow.
- **Phase 2 (ML):** model (XGBoost/LightGBM) → predicts probability of failure; heuristic as fallback.

🅲 Heuristic Scoring (Procedure)

- Define features + weights: LOC, files changed, dep bump, coverage delta, test fails, security issues, past failures, hardware penalty, criticality multiplier.
- Compute raw score → normalize to 0–100.
- Apply modifiers (criticality factor, platform penalty, coverage bonus).
- Output: `risk_score` + reasoning list.

📄 Deliverable: `risk-decision.json` (risk_score, top_contributors, recommended_strategy, suggested_pace)

🅳 Strategy Mapping

- Buckets:
- 0–30 → Rolling
- 31–70 → Canary
- 71–100 → Blue-Green
- Pace schedules:
- Rolling → large batch steps
- Canary → 10 → 25 → 50 → 100 (windowed)
- Blue-Green → Green validation before cutover
- Overrides:
- Critical CVE → force Blue-Green
- DB schema / kernel driver → Blue-Green + manual approval
- Hotfix → smaller canary + close monitoring
- Per-OS:
- ARM firmware more conservative

- Windows drivers +10 risk
- Linux → default mapping

📄 Deliverable: `strategy-mapping-policy.yaml`

🅔 Parameterizing Thresholds

- Collect historical builds + outcomes.
- Backtest heuristics: compare predictions vs actual failures.
- Tune thresholds to minimize false negatives, keep false positives manageable.
- Define per-service tolerances + required healthy windows.
- Governance: get sign-off before enforcing.

📄 Deliverable: `thresholds-and-windows.md`

🅕 Policy Storage & Versioning

- **Policy-as-code (Git):** store `risk-weights.yaml`, `strategy-mapping-policy.yaml`, `thresholds.yaml`.
- **Policy Registry:** Jenkins pulls from Git or a Policy Service.
- **Audit Trail:** record policy version, decision, evidence in Artifactory.
- **ML Models (if used):** store in MLflow/artifact registry; reference in policy.

📄 Deliverable: versioned policy repo + decision-audit records

---

🅖 Jenkins Pipeline Integration

- Add `Decide Strategy` stage after artifact publish:
- Fetch `risk-inputs.json`
- Load policy bundle
- Compute risk_score & strategy
- Emit `risk-decision.json`
- Attach to Artifactory & notify Slack
- Branch pipeline to Rolling / Canary / Blue-Green deployment stages.
- Enforce manual approval for high-risk (e.g., score ≥85).

📄 Deliverable: Jenkins pipeline stage in shadow → enforced

🅗 Shadow, Calibration & Enforcement

- **Shadow mode:** log decisions for 10–20 releases.
- **Calibrate:** tune weights/thresholds weekly.

- **Canary enforcement:** enable for subset of services.
- **Full enforcement:** after governance sign-off.

📄 Deliverable: shadow reports + enforcement decision

🅘 ML Path (Future)

- Collect features (LOC, files, test fails, deps, CVEs, history, author reliability).
- Label releases: 1 = rollback/failure, 0 = stable.
- Train & evaluate ML model → deploy in shadow.
- Always retain heuristic fallback.

📄 Deliverable: model registry entry + policy bundle with model version

---

🅙 Validation & Success Criteria

- Every candidate artifact has `risk-decision.json` attached.
- Policy version recorded + auditable.
- Shadow FPR <10% before enforcement.
- Manual approval required for high-risk.
- Per-OS exceptions respected.
- Accuracy improves over 3-month review cycle.

🅚 Immediate Actions (Checklist)

- [ ] Create `risk-inputs.json` template.
- [ ] Initialize policy repo with weights + mapping + thresholds.
- [ ] Add shadow `Decide Strategy` stage in Jenkins.
- [ ] Store decision evidence in Artifactory.
- [ ] Run 10–20 shadow runs + calibrate.
- [ ] Review + approve thresholds with SRE/leadership.

🔗 With this README, teams can implement Step 3 safely and iteratively, starting with heuristics and progressing to AI-driven strategy selection.