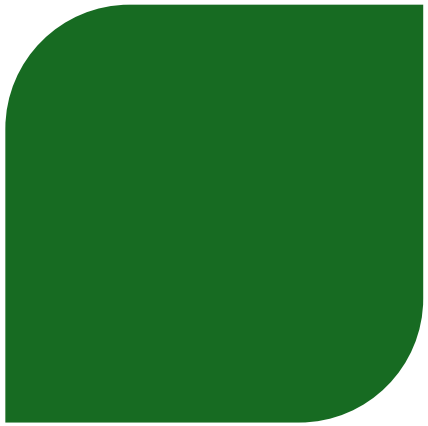
pixelplex

Building Quality into LLM-powered applications

What is an LLM-powered application

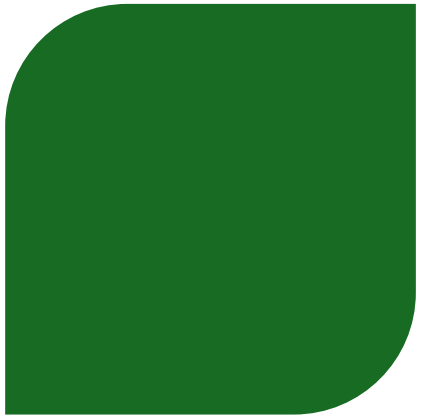
CODE WRITTEN LARGELY BY AI



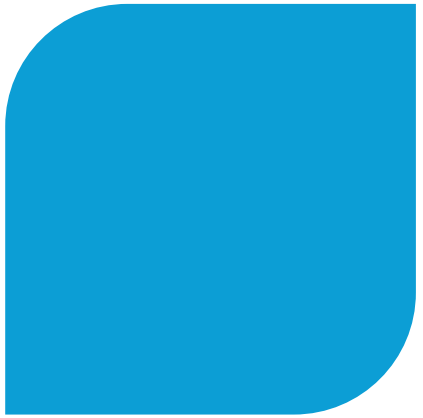
APPLICATION THAT USES AI WIDELY FOR OUTPUT

Future Proofing Your Build

UNDERSTANDING SOFTWARE DEVELOPMENT IS CHANGING

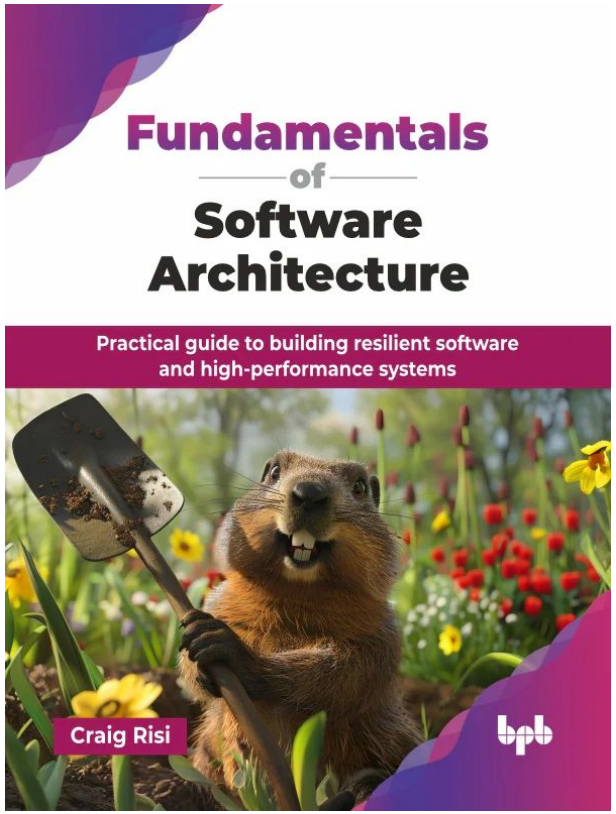


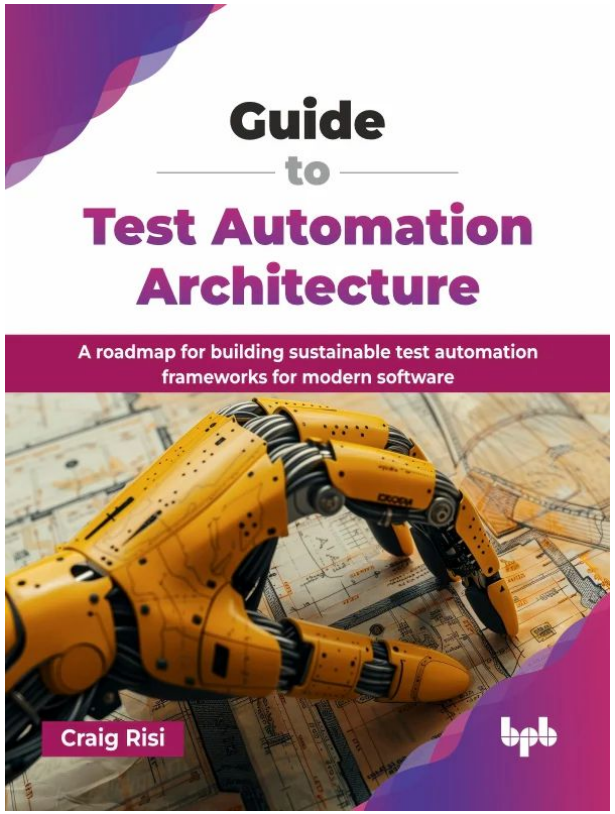
CHANGING THE LENS ON DEFINING QUALITY



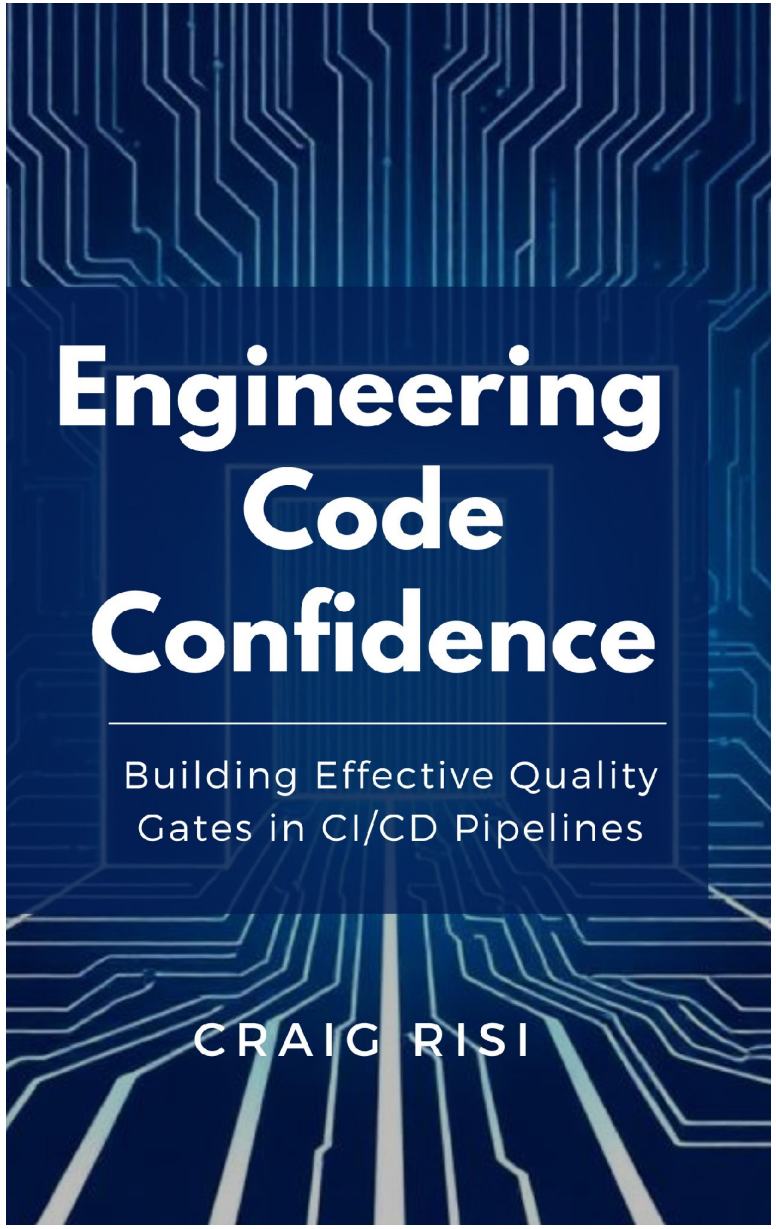
STRATEGIES FOR BUILDING QUALITY AND TESTING INTO AI-DRIVEN DEVELOPMENT


QUAY FOUR





www.craigrisi.com



www.craigrisi.com


PAST
FUTURE
NOW

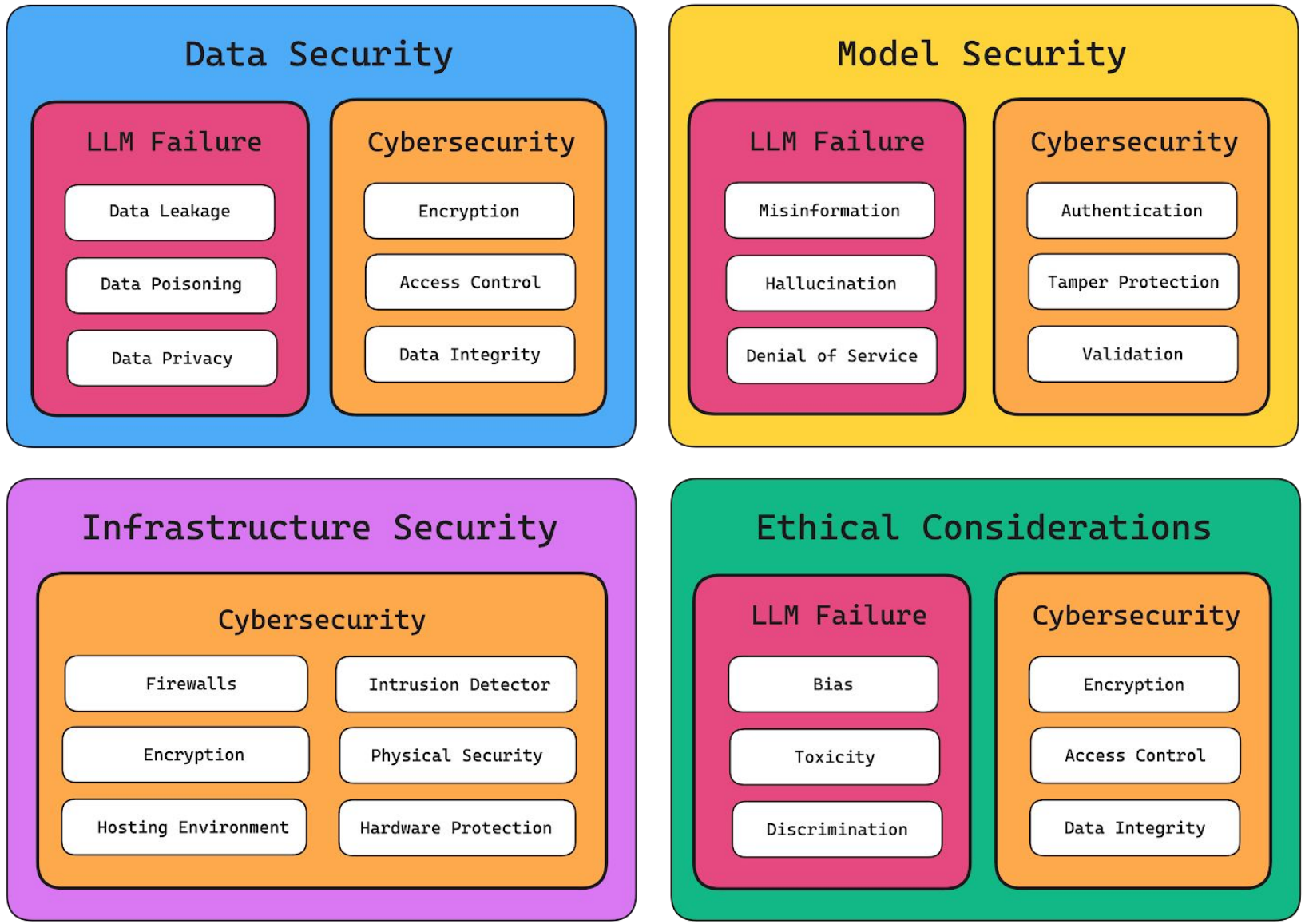
	AI Projects	Sofware Development Projects
Nature of Work	Exploration, discovery, probabilistic outcomes.	Building deterministic, functional systems
Uncertainty	High (depends on data quality, model feasibility)	Low (requirements defined upfront)
Iteration	Frequent revisiting of data, models, and evaluation	Iteration exists (e.g., Agile), but less exploratory.
Output	Insights, predictions, trained models	Functional software (e.g., applications).
Success Metrics	Accuracy, precision, recall; subjective evaluation	Binary: software works per specifications or not.
Data Dependancy	Central (requires preprocessing/exploration)	Secondary (data as input/output).

Challenges working with LLMs

Unique Challenges of modern LLM-powered Applications

- Non-deterministic outputs
- Context management
- Data quality
- Evaluation difficulty
- Integration complexity
- Cost Optimization

4 Pillars of LLM Security




Redefining Quality

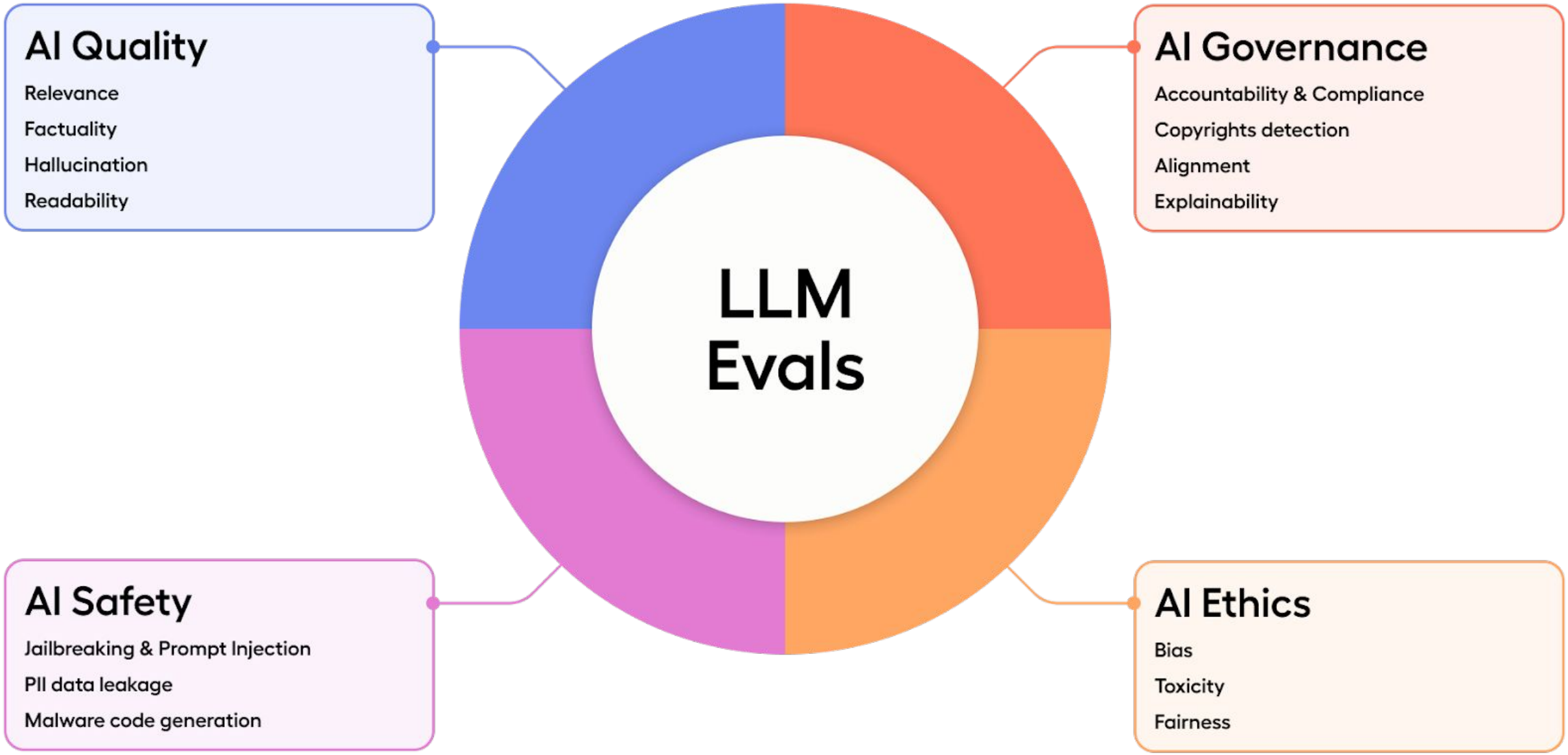
Changing the Dimensions of Quality

Accuracy: Does the model produce correct or factual results?

Reliability: Consistent behaviour across inputs and contexts.

Safety & Ethics: Avoiding harmful, biased, or unsafe outputs.

Performance: Speed, scalability, and cost efficiency.


AI Quality
Relevance
Factuality
Hallucination
Readability
AI Governance
Accountability & Compliance
Copyrights detection
Alignment
Explainability
LLM Evals
AI Safety
Jailbreaking & Prompt Injection
PII data leakage
Malware code generation
AI Ethics
Bias
Toxicity
Fairness


Unlocking LLM Potential

Strategies for Building Quality

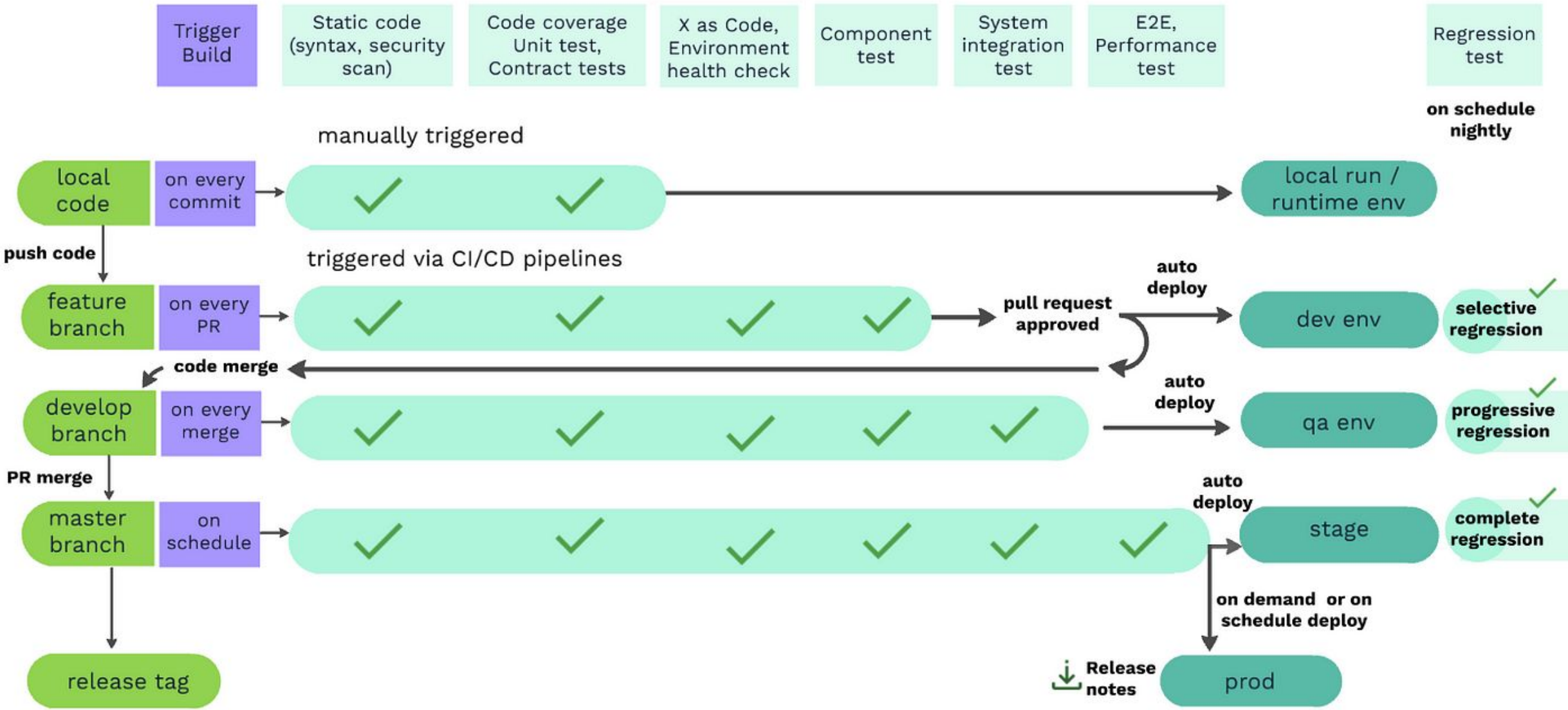
ENGINEERING PRINCIPLES

TESTING & EVALUATION FRAMEWORKS

SAFETY & ETHICS CHECKS

OBSERVABILITY & FEEDBACK LOOPS

GUARDRAILS & FAIL-SAFES


Trigger Build
Static code (syntax, security scan)
Code coverage Unit test, Contract tests
X as Code, Environment health check
Component test
System integration test
E2E, Performance test
Regression test
on schedule nightly
manually triggered
local code
on every commit
local run / runtime env
push code
triggered via CI/CD pipelines
feature branch
on every PR
pull request approved
auto deploy
dev env
selective regression
code merge
develop branch
on every merge
auto deploy
qa env
progressive regression
PR merge
master branch
on schedule
auto deploy
stage
complete regression
on demand or on schedule deploy
release tag
Release notes
prod

Testing LLM-powered applications

DATA QUALITY TESTING

TEST-DRIVEN DEVELOPMENT (STRICT UNIT TESTING)

WORK WITH GOLDEN DATASETS

LINTING AND QUALITY CONTROL

SECURITY AND PERFORMANCE TESTING

EARLY MONITORING AND OBSERVABILITY

AI Testing Strategies & Methodologies


Questions