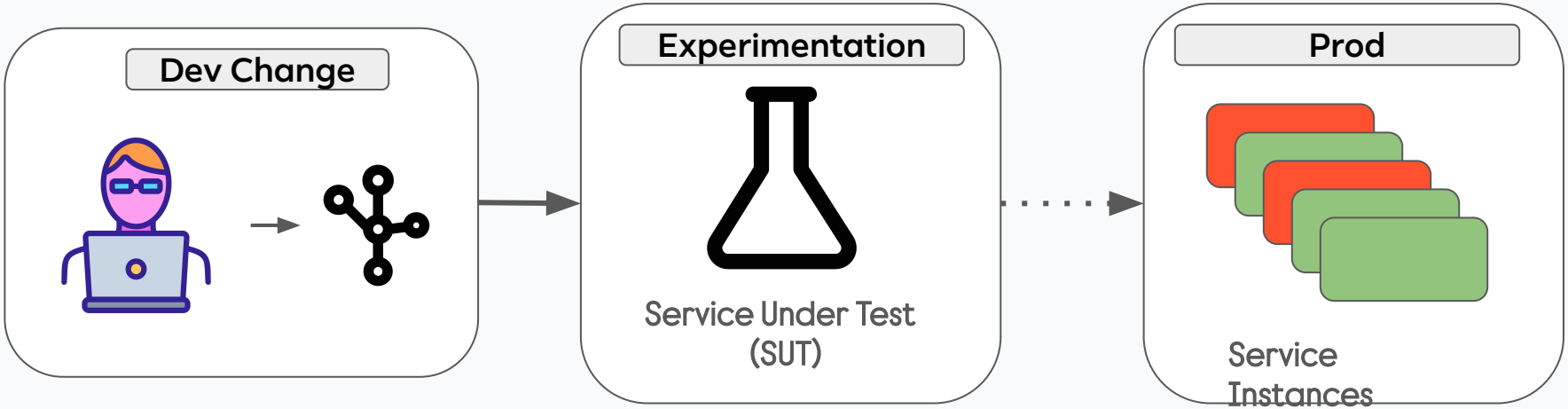
Experimenting with Fire

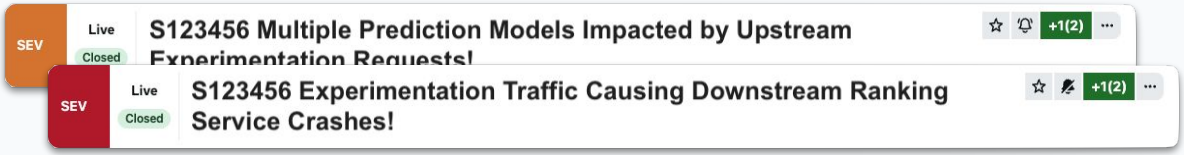
Protecting Production from High-Velocity Changes at Scale

Shripad Nadgowda, Canturk Isci
Meta Recommendation Systems (MRS) Infra

Meta

Dev Changes Crashed Production - Before Production Deployment







Outline for the talk

CHAPTER 01 | Experimentation in Prod

CHAPTER 02 | Experimentation Platform

CHAPTER 03 | Risks and Failure Modes

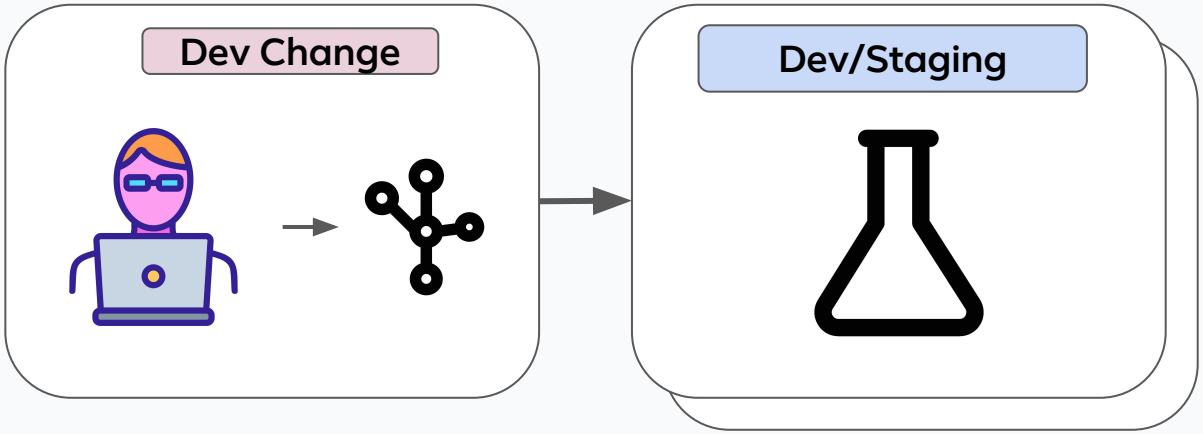
CHAPTER 04 | Building the Guardrails

CHAPTER 05 | Lessons from Production



Chapter 1: Experimentation in Prod

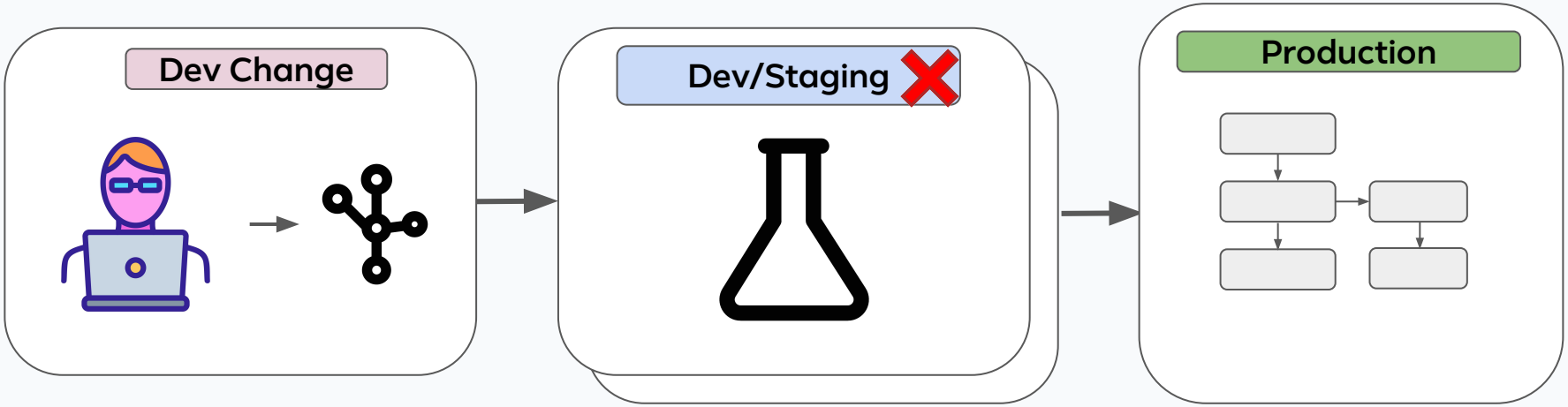
Experimentation



- A/B, A/A, Integration testing
- Diff/Land time validation
- Release management
- Debugging, Bisect, New HW



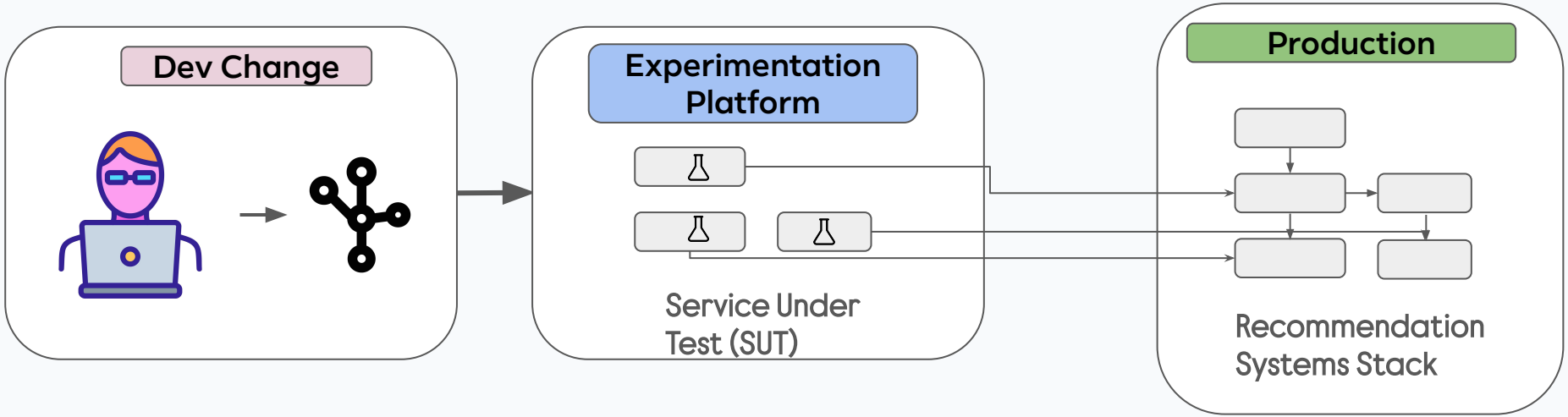
Experimentation - in Prod!



- **Full graph cannot be replicated**: maintaining a production-equivalent staging stack does not scale.
- **Specialized capacity is scarce:** Accelerator fleets, 1000s of models, impractical to duplicate.
- **Realistic Signal, State drifts:** Feature and content stores change continuously.



Experiment:SUT → Deps:Prod Connection



Isolate SUT execution while keeping production-connected dependencies

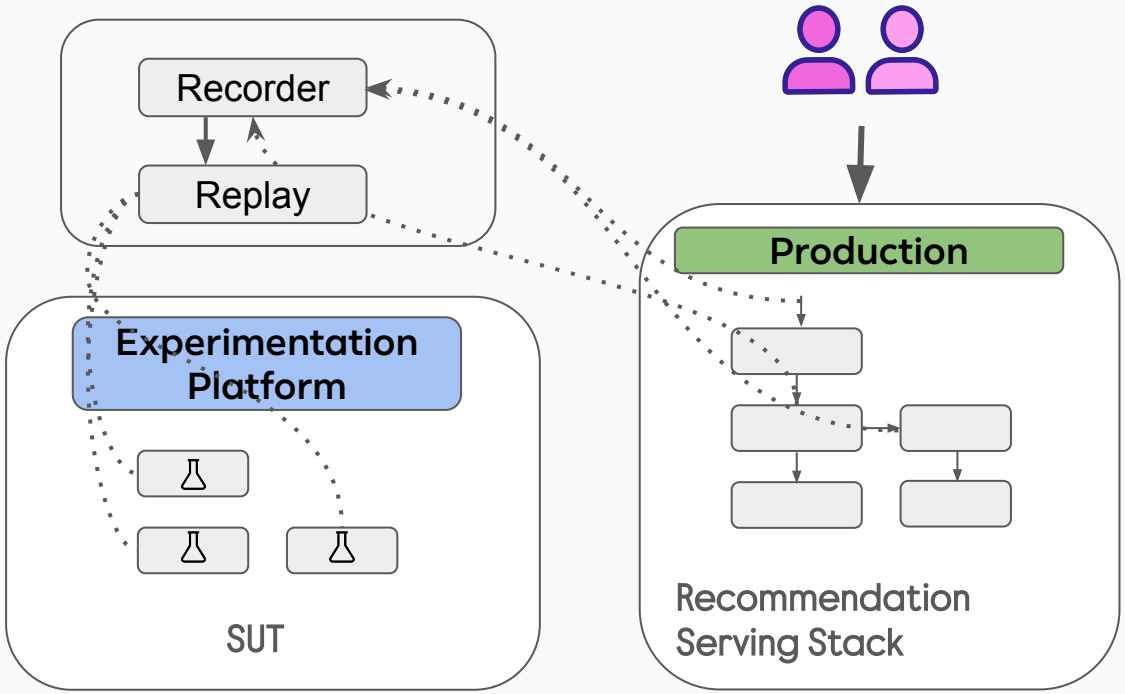




Chapter 2: Experimentation Platform

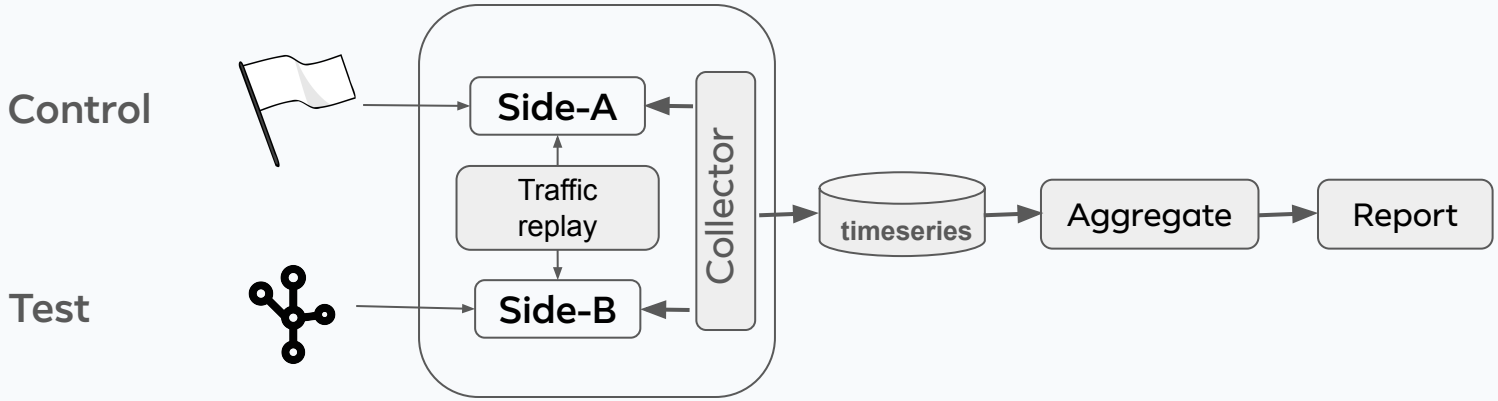
Record & Replay

Record and Replay





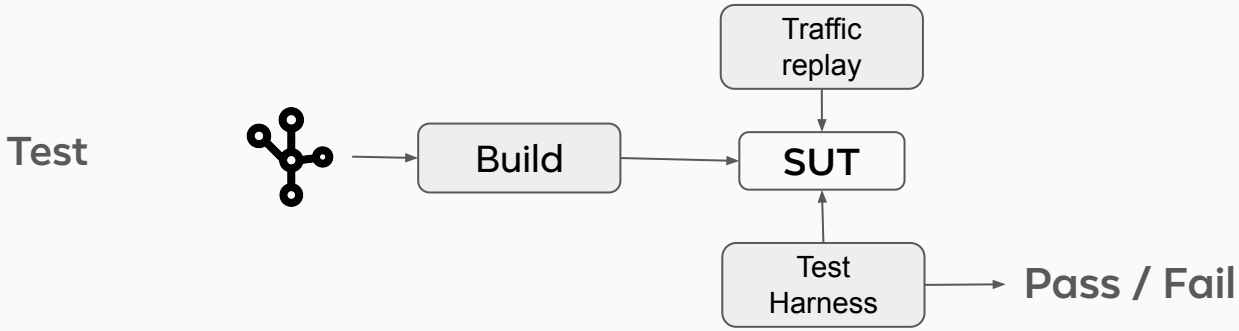
Experiment Topologies (1/2): A/B Comparative



Matched control and test counters are aggregated across replay trials to detect regressions



Experiment Topologies (2/2): Integration Pass/Fail

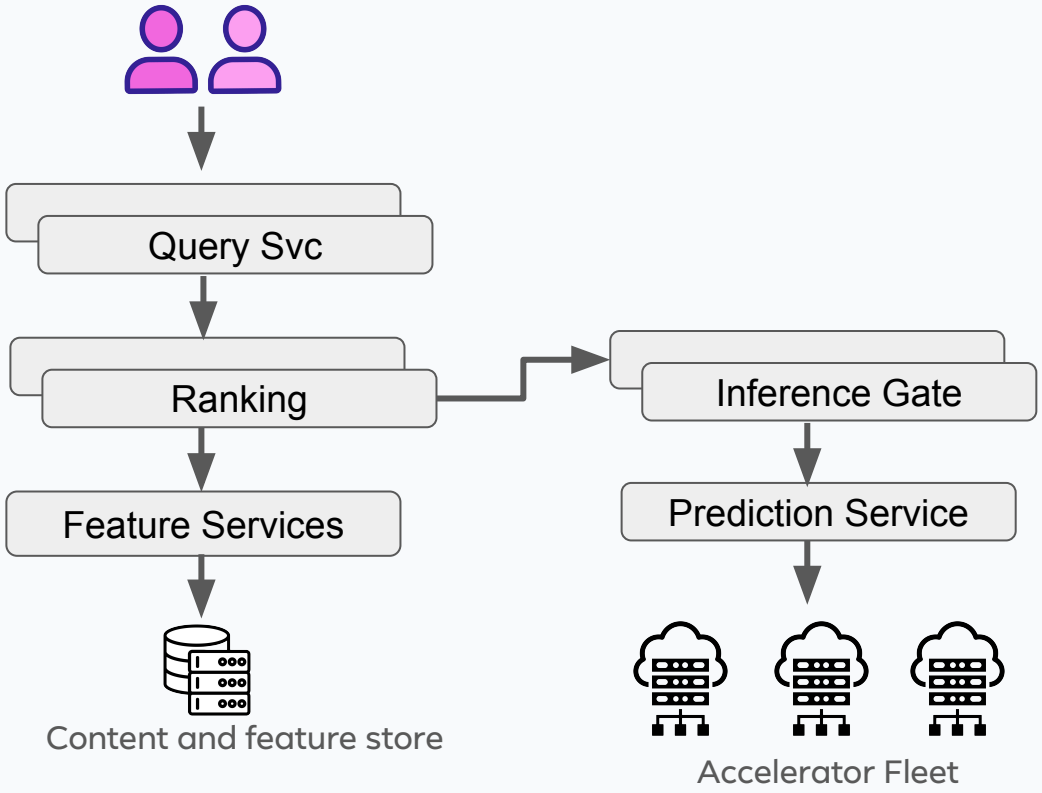


Programmable service-specific checks during and after replay



What exactly are we testing ?

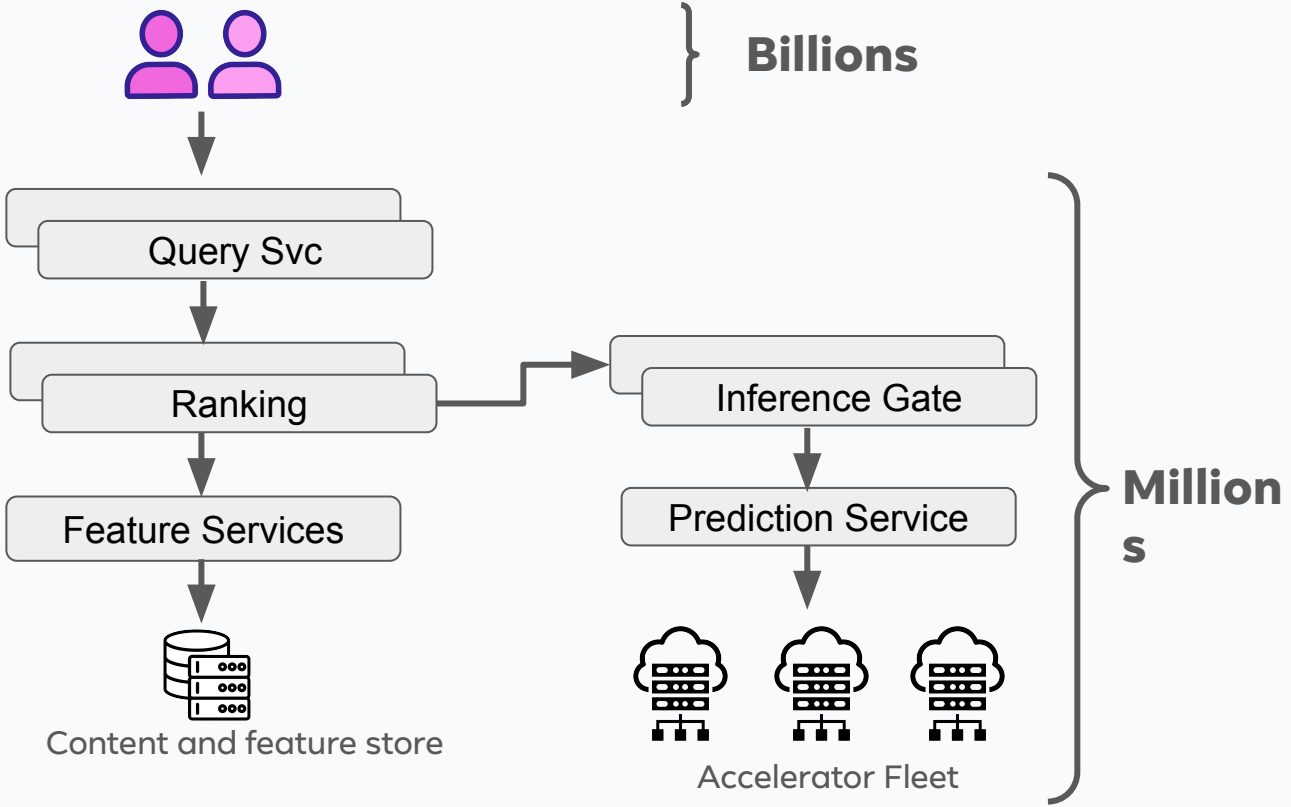
Recommendation System





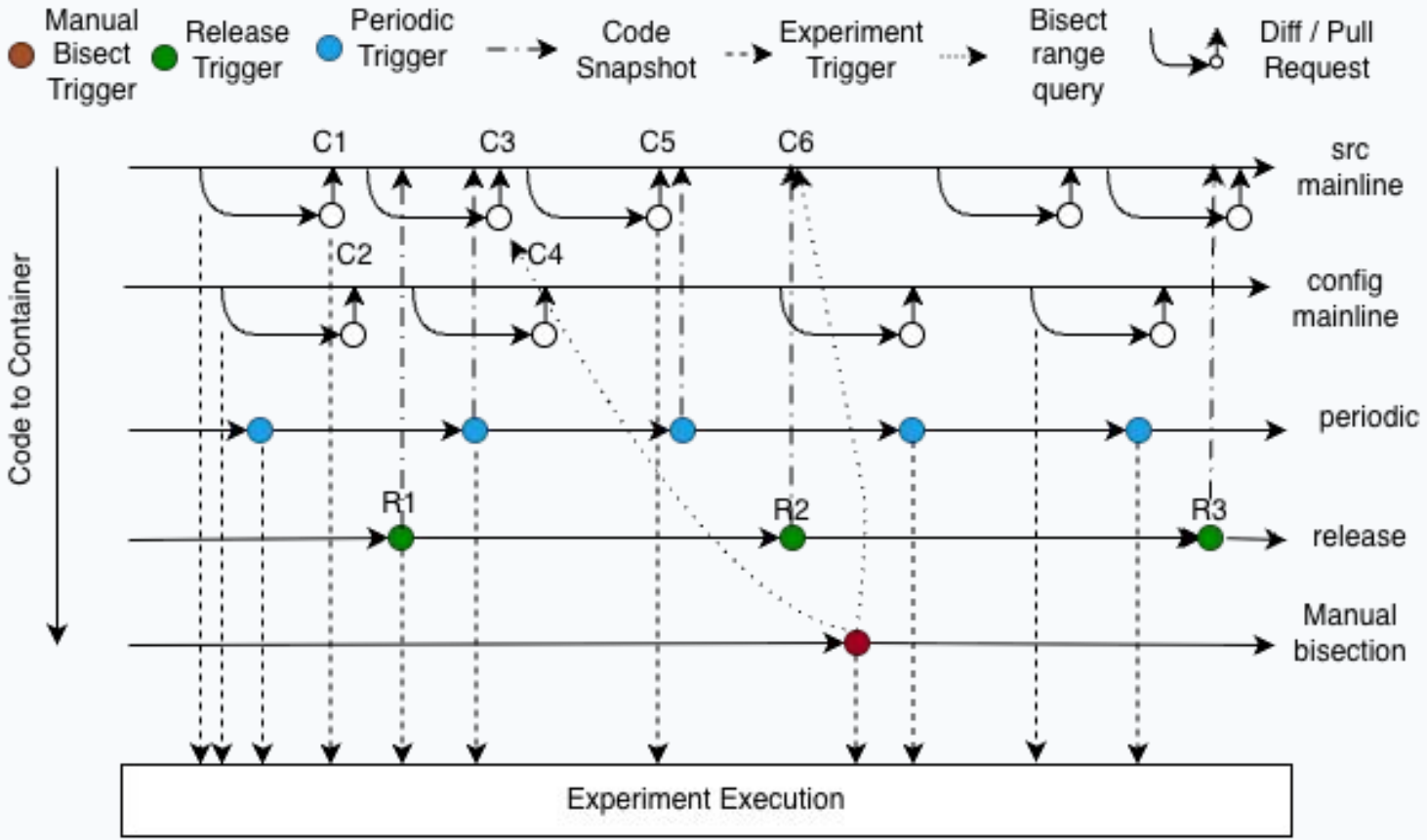
What exactly are we testing ?

Hyperscale Recommendation System





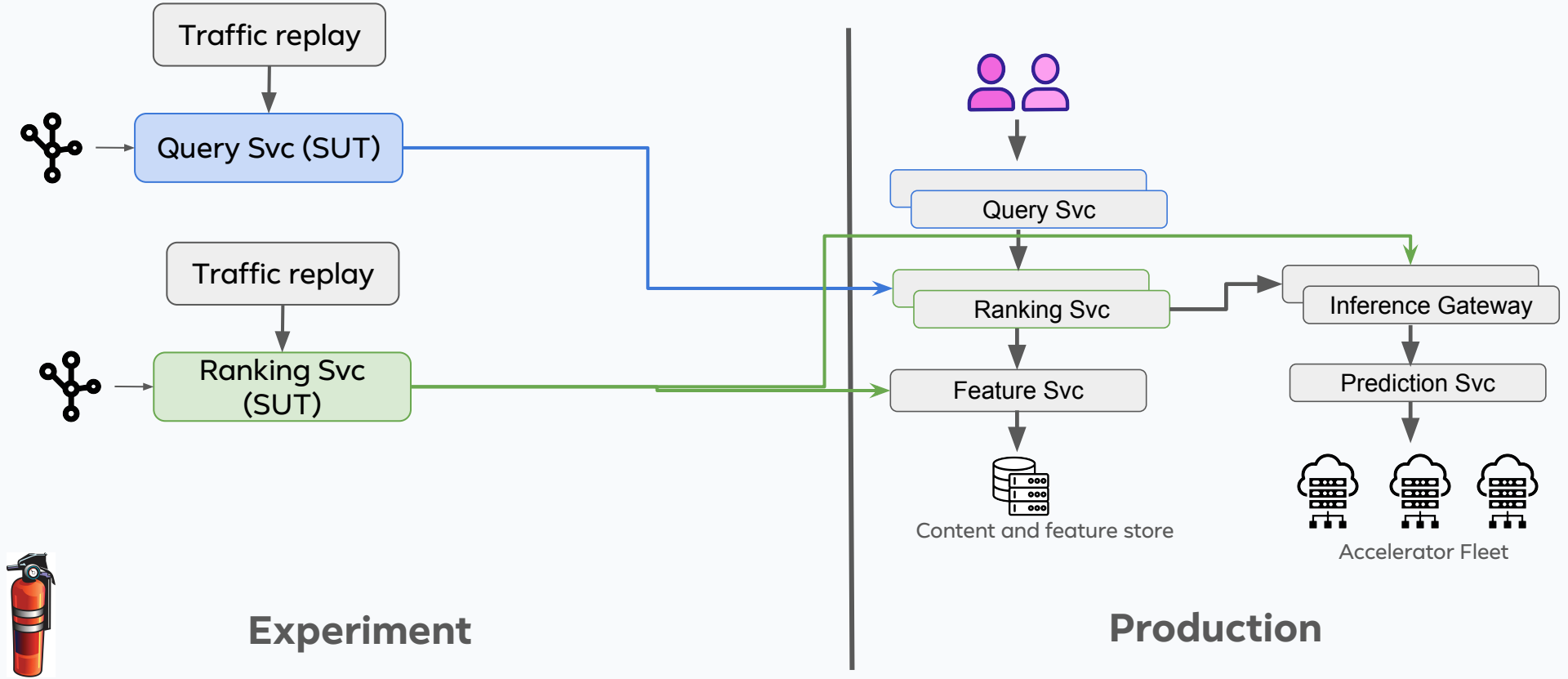
Experiment Fanout :o



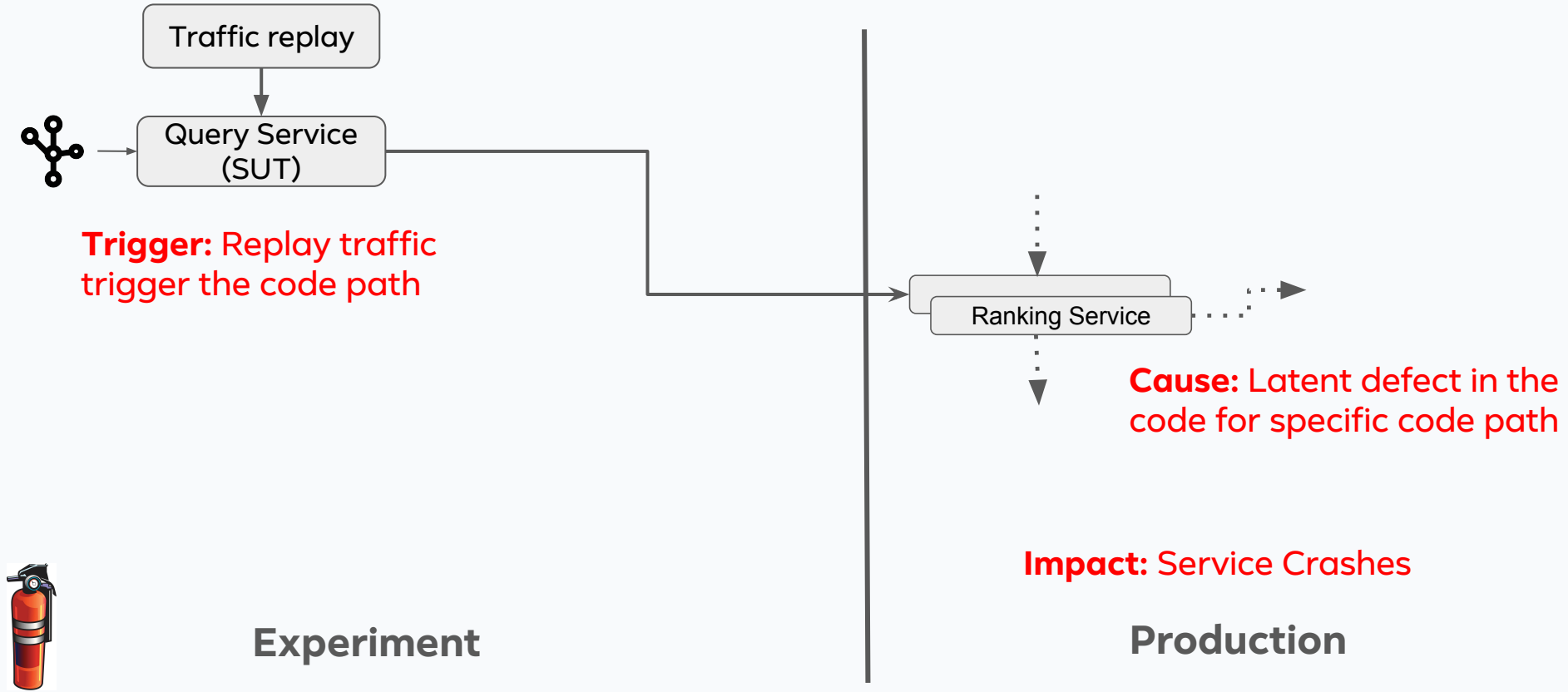


Chapter 3: Production Connectivity: Risks and Failure Modes

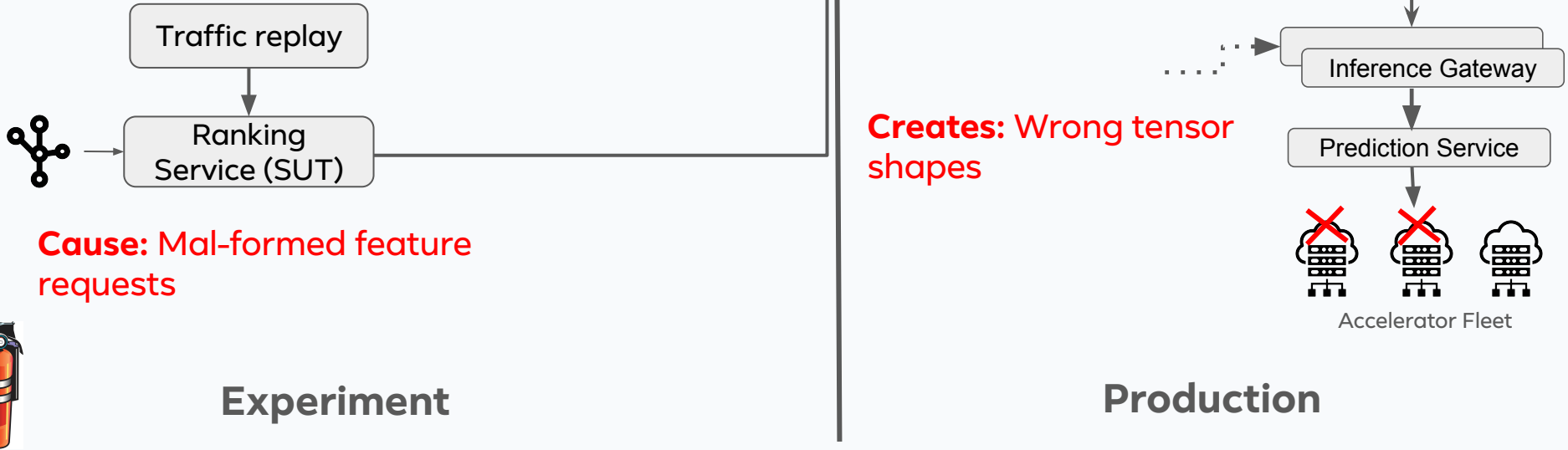
Production Connection



Failure Modes



Failure Modes



Hmm..

RISKS		CONSEQUENCE
Replay traffic reaches shared production dependencies	→	Shared serving capacity can be affected
Failure appears outside the SUT	→	Failure attribution is hard
Fault reaches many workers	→	Small traffic volume can cause broad impact
Old replay keeps dependency state active	→	Cleanup or model reaping is delayed



Chapter 4: Building the Guardrails



Now, the Guardrails

GUARDRAIL 01

Isolation

Route replay traffic away from shared production dependencies

GUARDRAIL 02

Containment

Bound exposure to a designated worker subset

GUARDRAIL 03

Dependency Monitors

Detect sustained downstream impact and stop the experiment



Selective Isolation

Remove Shared Exposure

Selective Isolation

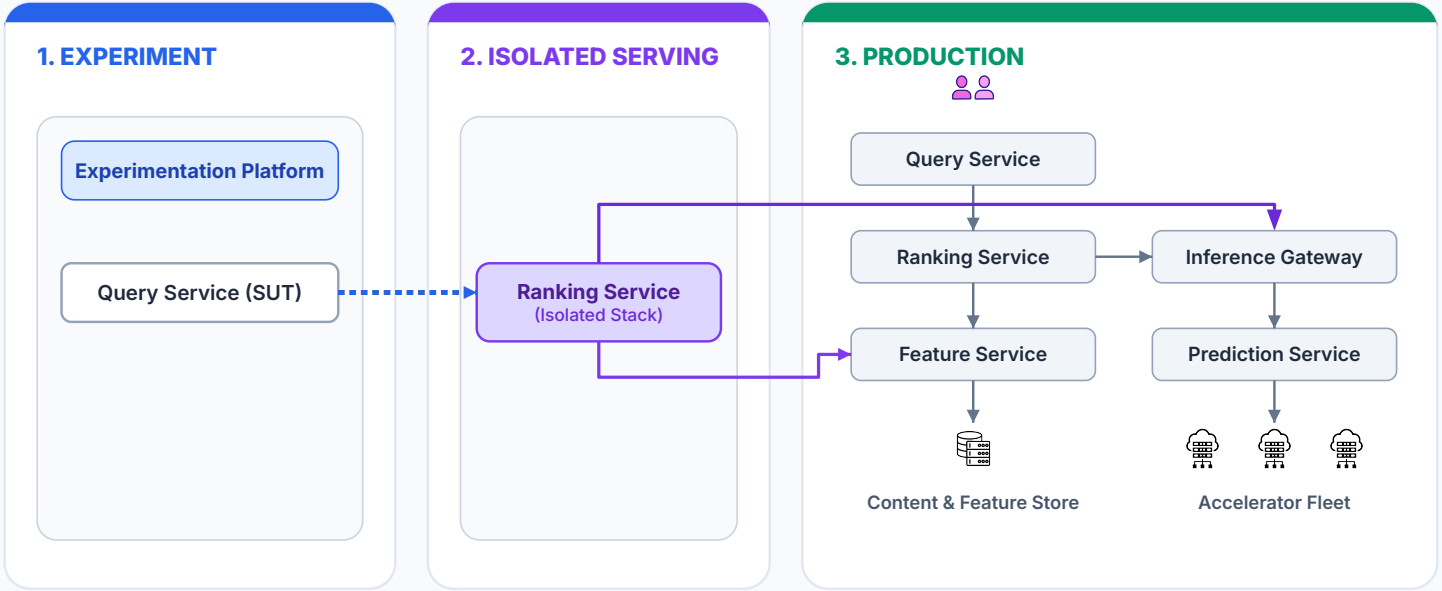
Route replay traffic away from shared production dependencies to dedicated isolated tiers

FaithFul and Practical

Isolation Criteria

Use isolation when dependency state can be reproduced, kept current, and provisioned with practical capacity





Containment

BOUND EXPERIMENT EXPOSURE

Exposure Control

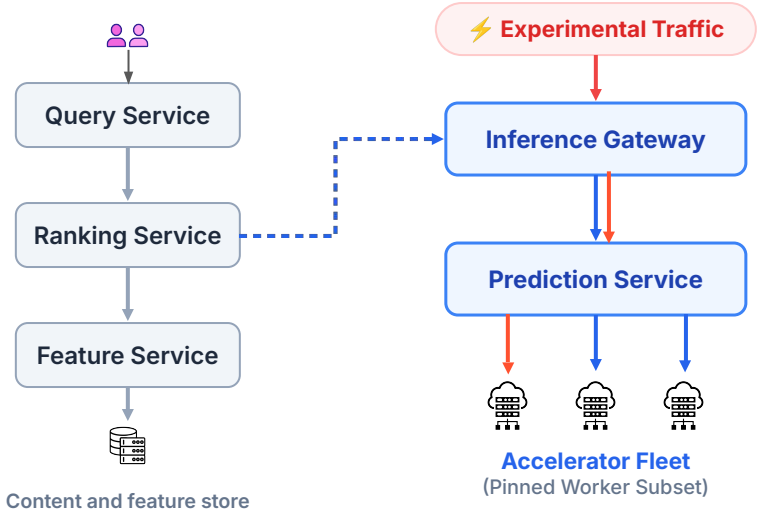
Bound experiment exposure when full dependency isolation is impractical.

TRAFFIC PINNING

Designated Subset Routing

Traffic pinning: Route experimental calls to a designated worker subset instead of the entire fleet.

CONTAINMENT ARCHITECTURE & WORKFLOW





Dependency Monitors

HEALTH SIGNAL MONITORING

Observe & Stop

Observe health signals from shared downstream dependencies and stop on sustained violations.

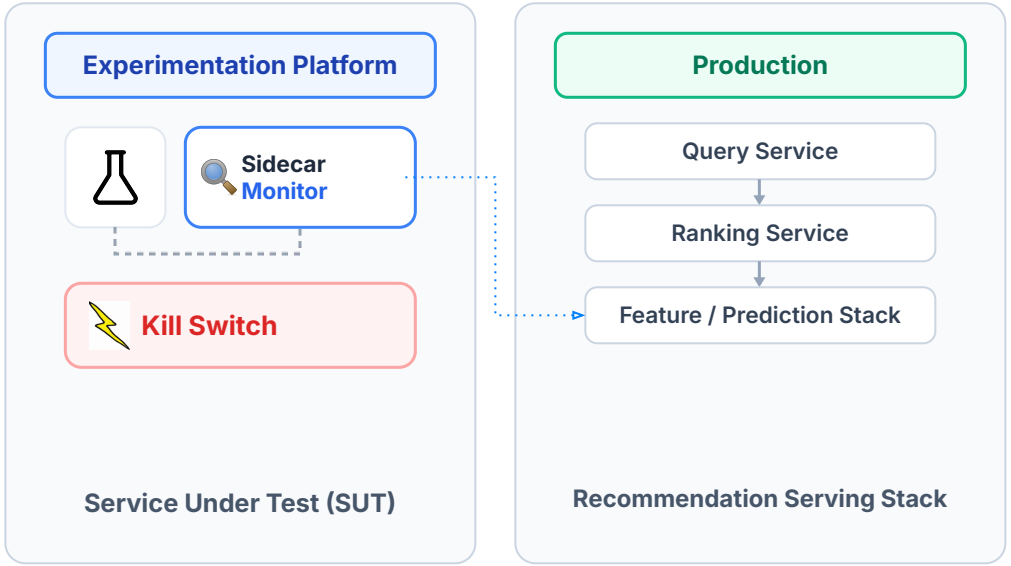
DETECTOR TYPES

Dependency-health: Track overall service health, such as worker crashes or restarts.

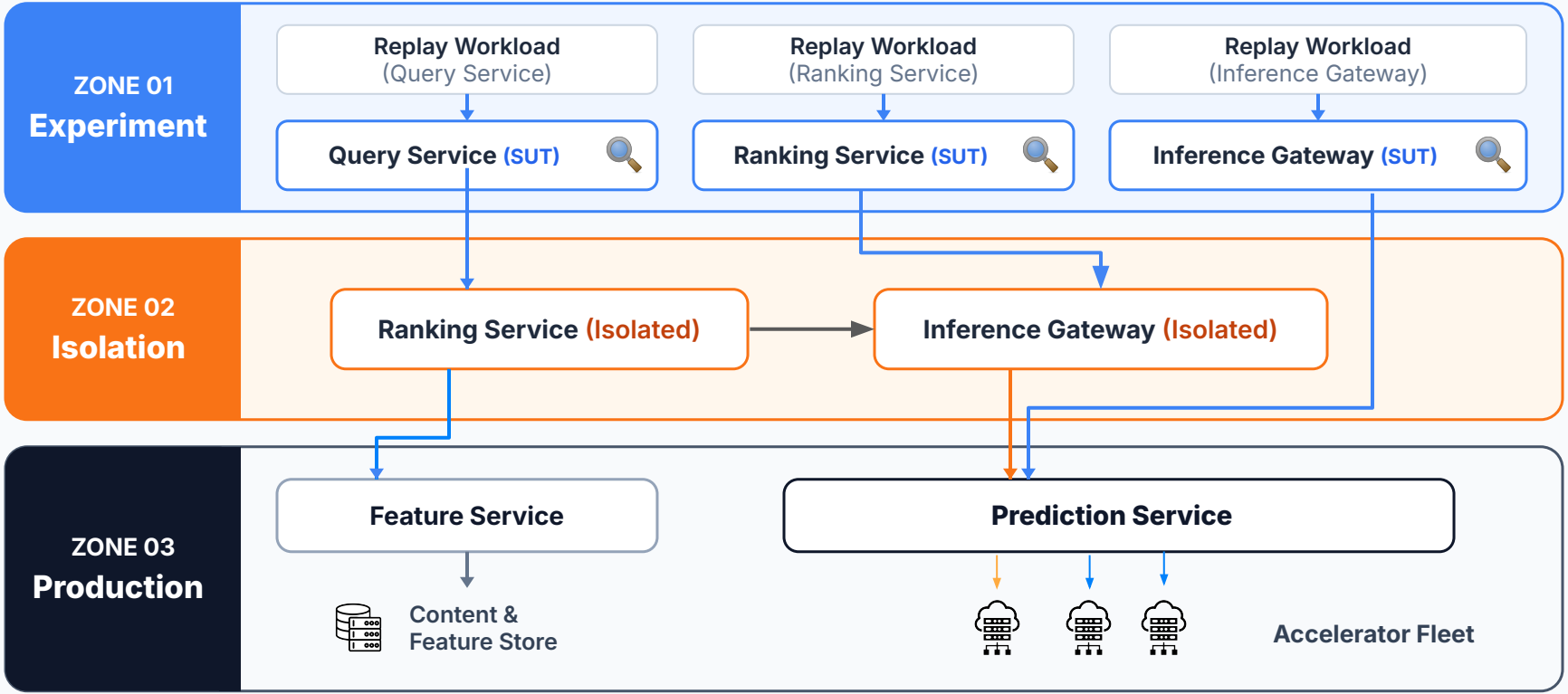
Experiment-attributed: Link downstream errors to the specific experiment that caused them.



DEPENDENCY MONITORING ARCHITECTURE



Putting them all together







Chapter 5: Lessons from Production

Lessons Learned

Core Principle: No single guardrail fits every dependency.

01 | AUTOMATED CONTAINMENT

Attribution is a prerequisite for automated containment.

02 | BLAST RADIUS

Blast radius is measured by workers reached—not traffic volume.

03 | PROACTIVE ISOLATION

Monitoring reacts to failures; isolation and pinning limit exposure beforehand.

04 | EXPERIMENTATION CONTROL

Avoiding unnecessary experiments is also a reliability mechanism.

05 | FAULT LIMITATIONS

Guardrails contain faults; they do not eliminate them.



Thank you !

