Netflix Interview Prep - Staff Engineer, Commerce ML Data Products

Interview Process (What to Expect)

Netflix usually does:

1. Recruiter call (30 min) - logistics, comp, basic fit
2. Hiring manager (45-60 min) - your background, system design chat
3. Technical deep dive (60-90 min) - architecture, distributed systems
4. Cross-functional partner (45-60 min) - how you work with ML teams
5. Culture interview (45-60 min) - Netflix values alignment
6. Team rounds (2-3 x 45-60 min) - technical depth, domain knowledge

---

Your Main Story

Use this in every interview:

"For the past 4 years at Walmart, I've been building infrastructure that looks a lot like what ML systems need: real-time feature computation, low-latency serving, Kafka streaming, developer platforms. The patterns are the same whether you're scoring fraud or serving model predictions. I've just been doing it for fraud prevention instead of Commerce optimization. Now I want to apply these patterns to actual ML models at Netflix."

Talking Points: Your Experience → Netflix's Needs

1. Real-time feature computation

What you built: Distributed counting system at Walmart

- Processes millions of events/second with sub-second latency
- Basically a feature store before I knew that term
- Stack: Kafka ingestion → distributed computation → cache → REST API
- Solved: late arrivals, exactly-once semantics, backfills, schema changes

Why it matters to Netflix: Commerce features (user behavior, payment history, device signals) need the same architecture - real-time computation with strong freshness guarantees.

Questions they'll ask:

- "How do you handle late-arriving events?" → Windowing with grace periods, out-of-order processing, idempotency
- "How do you ensure exactly-once?" → Kafka offsets, transactional writes, deduplication
- "How do you backfill features?" → Separate batch pipeline, same event log, temporal consistency

2. Kafka streaming

What you built: Multiple Kafka systems handling Black Friday traffic (10x normal load)

- Event sourcing patterns - Kafka as source of truth
- Solved: partition strategies, consumer lag, topic retention, schema registry

Why it matters to Netflix: Their Commerce events (purchases, identity, payments) have the same characteristics: high volume, ordering requirements, multiple consumers.

Questions they'll ask:

- "How do you partition topics?" → Depends on use case: user_id vs session_id vs transaction_id, handling hotkeys
- "How do you handle backpressure?" → Consumer scaling, rate limiting, circuit breakers
- "How do you monitor lag?" → Real-time metrics, alerts, automated scaling

3. Low-latency serving

What you built: Account Risk API at Walmart

- p99 latency under 50ms, millions of requests/day
- 99.99% uptime (about 4 minutes downtime per month)
- Stack: Node.js, Kafka, Memcached, distributed tracing
- Optimizations: connection pooling, circuit breakers, cache warming

Why it matters to Netflix: Commerce ML needs same guarantees - fraud detection, payment optimization must be real-time.

Questions they'll ask:

- "How do you get sub-100ms?" → Hot path optimization, pre-computation, caching, async where possible
- "How do you handle cache invalidation?" → TTL-based, event-driven, read-through patterns

- "How do you ensure 99.99% uptime?" → Multi-region, health checks, circuit breakers, graceful degradation

4. Developer platforms

What you built: Internal platform at Walmart

- Engineers write logic, platform handles infrastructure
- 60% reduction in time-to-production
- Features: API for data access, job scheduling, secrets, monitoring dashboard

Why it matters to Netflix: ML engineers need easy access to Commerce data - focus on models, not infrastructure.

Questions they'll ask:

- "How do you design self-service platforms?" → API-first, clear abstractions, good docs, support
- "How do you balance flexibility vs guardrails?" → Opinionated defaults, escape hatches for power users
- "How do you measure success?" → Time-to-production, adoption rate, support tickets

5. Operational reliability

What you built: All my Walmart systems run at 99.99% uptime

- Full observability: Splunk, Grafana, distributed tracing, automated alerts
- Three levels: real-time dashboards, SLA alerts, post-incident analysis

Why it matters to Netflix: Commerce ML can't have downtime - if payment models are down, Netflix loses revenue.

Questions they'll ask:

- "How do you debug latency spikes?" → Distributed tracing, correlation with deploys/traffic, cache analysis
- "How do you prevent cascading failures?" → Circuit breakers, bulkheads, rate limiting, graceful degradation
- "What's your on-call philosophy?" → Runbooks, blameless post-mortems, automation to reduce toil

---

Addressing Your Gaps (Turn Weaknesses into Strengths)

Gap: "You haven't used Spark or Flink"

What to say: "True - my distributed processing has been with Go and Kafka. But the principles are the same: partitioning, fault tolerance, state management, backpressure. I can learn Spark/Flink quickly - I've picked up frameworks throughout my career. What I bring is deep production systems thinking: designing for reliability, observability, scale. The framework is the easy part."

Gap: "You haven't worked with ML models"

What to say: "Right - I've built the infrastructure ML needs, not the models themselves. But that's what this role is: building the data substrate. I understand feature freshness, online/offline consistency, backfill, data quality - I've solved these at Walmart. What I'm excited to learn is how ML models consume data, so I can build even better products for Netflix ML engineers. I want to be the infrastructure expert who deeply understands ML needs."

Gap: "You don't know Netflix's internal frameworks"

What to say: "Every company has unique tooling - that's expected. I learn quickly by pattern-matching to what I know. When I see Netflix's frameworks, I ask: 'What problem does this solve? What are the tradeoffs?' Then map it to distributed systems principles I understand. I've done this at Walmart - dive into codebases, read docs, pair with engineers, contribute within weeks."

Stories for Behavioral Questions

"Tell me about driving technical direction across teams"

Story: Event Automation Platform

- Situation: Multiple teams at Walmart wanted different deployment workflows, no consensus
- My approach: Met with stakeholders, designed flexible platform with good defaults, built MVP and iterated
- Impact: Now handles 100+ deployments/week, reduced manual work 80%, became the standard
- Lesson: Start with user needs, design for flexibility, build trust through iteration

"Tell me about disagreeing with a decision"

Story: Kubernetes Migration

- Situation: Push to migrate everything to Kubernetes immediately at Walmart

- My position: Some services weren't ready, migration risk was high
- My approach: Proposed phased migration, built cost/benefit analysis, presented with data
- Outcome: Leadership agreed, we migrated incrementally, avoided major incidents
- Lesson: Disagree with data, propose alternatives, assume positive intent

"Tell me about operating in ambiguity"

Story: IntelDb Threat System

- Situation: Leadership said "we need real-time threat intelligence" but requirements were vague
- My approach: Talked to users to understand pain points, built prototype, iterated
- Challenge: Requirements kept changing as we learned
- Outcome: Shipped in 6 months, now processes 10M+ events/day
- Lesson: In ambiguity, bias toward action - prototypes clarify faster than specs

---

Netflix Culture Questions

Freedom & Responsibility

Q: "Tell me about a time you had freedom and the outcome"

"At Walmart, I had full ownership of our feature aggregation architecture. Leadership trusted me on tech stack, design, deployment. I chose Go for performance, Kafka for streaming, Memcached for serving. Risk: if I was wrong, it impacts all of ecommerce. Outcome: system ran at 99.99% uptime for 4 years, became the model for other teams. Freedom works for me because I balance boldness with risk mitigation - prototype, measure, iterate."

Candor

Q: "Tell me about giving difficult feedback"

"A senior engineer was building a pipeline that wouldn't scale. I told them directly: 'This design has a single point of failure. If this goes down, we lose all data.' They were defensive. I showed latency data, walked through failure scenarios. They redesigned it - now one of our most reliable systems. Candor is kinder than letting someone fail. But it has to come from respect and wanting them to succeed."

Innovation

Q: "How do you stay current with tech?"

"I'm a top 2% Cursor AI user - I don't just read about AI, I use it daily. I follow Netflix's tech blog, Databricks, MLOps communities. I read papers on feature stores, distributed ML. But I'm pragmatic - adopt tech when it solves real problems, not for hype. At Walmart, I pioneered LLM-assisted development because it made my team 40% more productive."

Context, Not Control

Q: "Tell me about empowering others"

"I built a platform at Walmart that gives engineers full control. I don't review every change - I provide context through docs and guardrails. Example: An engineer wasn't sure about rollout strategy. Instead of telling them what to do, I asked: 'What's the blast radius if this fails? How will you monitor it?' They figured it out. My job isn't to make every decision - it's to provide context so others can make good decisions."

System Design Questions You'll Get

"Design a real-time feature store for Netflix Commerce"

Requirements clarification:

- What types of features? User behavior, payments, device signals?
- Latency requirements? <100ms for online inference?
- Scale? Millions of users, billions of events/day?
- Consistency? Strong vs eventual?

High-level design:

- Ingestion: Kafka topics for different event types
- Computation: Stream processing (Flink) for real-time aggregations
- Storage: Dual-write to batch (S3/Parquet) and online (Cassandra/DynamoDB)
- Serving: REST API with Redis cache for low latency
- Orchestration: Airflow for batch backfills, lineage tracking

Deep dives:

- Online/offline consistency: same computation logic for batch and streaming
- Schema evolution: Avro/Protobuf with schema registry
- Monitoring: feature freshness, data quality, SLA dashboards

- Backfill: batch job reading from Kafka retention or S3 archive

Tradeoffs:

- Lambda (batch + streaming) vs Kappa (streaming only)
- Precomputed vs on-demand features
- Strong vs eventual consistency

Your edge: "I've built this at Walmart - let me walk through what worked and what I'd do differently."

"Debug a sudden drop in feature freshness"

Immediate triage:

- Check dashboards: Kafka consumer lag? Processing errors?
- Recent deployments: code change or config update?
- Dependencies: upstream data sources healthy?

Hypothesis generation:

- Consumer lag → processing speed or partitioning issue
- Error spike → schema change or bad input
- Upstream delay → dependency failure

Investigation:

- Distributed tracing: which component is slow?
- Logs: exceptions or warnings?
- Metrics: CPU, memory, network bottlenecks?

Mitigation:

- Short-term: scale consumers, pause non-critical jobs, alert ML teams
- Long-term: fix root cause, add preventive monitoring

Your edge: "At Walmart, I debugged similar issues where schema changes broke consumers."

"Design A/B testing for ML models"

Requirements:

- Traffic split (e.g., 95% control, 5% treatment)
- Consistent user assignment (same user sees same model)
- Real-time metrics (latency, errors, business KPIs)
- Rollback if treatment underperforms

Design:

- Model registry: store versions with metadata
- Traffic router: hash user_id to assign variant
- Serving: parallel inference (both models score, log results)
- Metrics: real-time aggregation by variant
- Decision engine: automated rollback on SLA violations

Tradeoffs:

- Parallel inference (higher cost) vs shadow mode (no user impact)
- Real-time metrics (complex) vs batch analysis (delayed)

Your edge: "At Walmart, we built similar A/B infrastructure for rule changes."

Questions to Ask Them

For hiring manager:

1. What are the top 2-3 technical challenges Commerce ML is facing?
2. How does Commerce ML partner with ML Platform? Where are handoffs?
3. What does success look like in first 6 months?
4. How do you balance technical debt vs new features?

For ML engineers:

1. What's the biggest pain point with ML data today?
2. If you could fix one thing about feature engineering, what would it be?
3. How do you debug model performance issues? Is data quality a common blocker?

For the team:

1. What's a recent technical decision you debated? How did you resolve it?
2. How do you balance building new data products vs maintaining existing ones?
3. What's on-call like? How often do production issues happen?

For everyone:

1. What do you love about working at Netflix?
2. How does freedom & responsibility show up day-to-day?

Compensation

Netflix range: $230K - $960K (salary only, no bonuses, choose stock vs cash)

Your target: $300K - $500K+ total comp

What to say:

- "I'm currently at Walmart making [your comp]. Netflix's range aligns with my expectations."
- "I'm looking for $300K-$400K base + stock, total comp around $400K-$500K."
- "I'm flexible on salary vs stock split - I want to be invested in Netflix's success."
- "Most important is scope of impact, technical challenges, team culture. If those align, we'll find a number that works."

Don't undersell yourself - you have extensive experience and Staff-level expertise.

Final Prep

Before each interview:

- Review job description - map your stories to their needs
- Check Netflix Tech Blog - mention recent posts
- Prepare 2-3 questions for the interviewer
- Test video/audio
- Have your opening statement ready

Day before:

- Re-read your resume - be ready to go deep on anything
- Review your system designs
- Get good sleep

Day of:

- Log in 5 minutes early
- Have water nearby
- Have pen and paper
- Be yourself

Remember

You've built production systems at Walmart scale for 4+ years. You know distributed systems, operational reliability, cross-functional work. You just need to translate your experience into "ML data products" language.

Your strengths:

- Deep production systems thinking (not learnable)
- Real experience at scale
- Know how to ship reliable systems

Your gaps:

- Spark/Flink (learnable)
- Netflix frameworks (learnable)
- Working with ML models directly (you'll learn on the job)

Be confident. Be curious. Be yourself.

You're not "trying to break into ML" - you've been building ML infrastructure, just not called that.

Good luck! 🚀