No “Big Bang” Theory: A Two-Way Data Synchronization Pattern

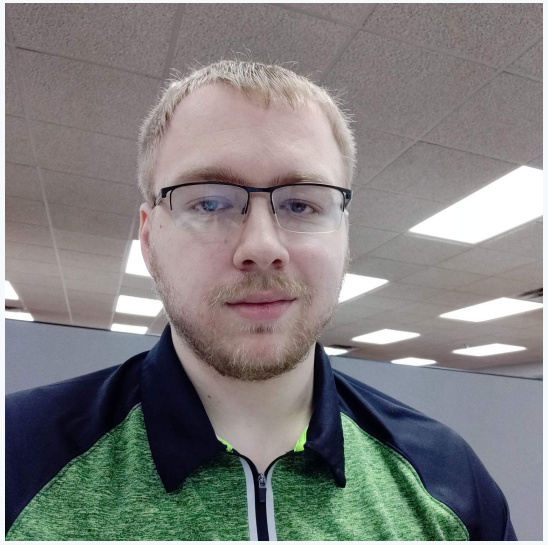
AWS Midwest Community Day 2026

CleanSlate TECHNOLOGY GROUP

Today's Speakers

Artie Sluka
Technical Manager
CleanSlate

Daniel Schnipke
Engineer
CleanSlate



Aaron Thomas
Senior Software Engineer
Byrider

The Problem

Why "Big Bang" migrations fail

CleanSlate TECHNOLOGY GROUP

The "Big Bang" Problem

CleanSlate TECHNOLOGY GROUP

“Big Bang” Migrations

- Typical modernization efforts: Multi-year (3+) projects, complete rewrites and tech stack overhauls
- No containers, no lift + shift, a whole presentation by itself on rationale
- Key issue: **flip the switch and hope it works**
- Keeps key stakeholders and executives up at night
- They would much rather build in small pieces and deliver incrementally (strangler pattern)

The "Big Bang" Problem

CleanSlate TECHNOLOGY GROUP

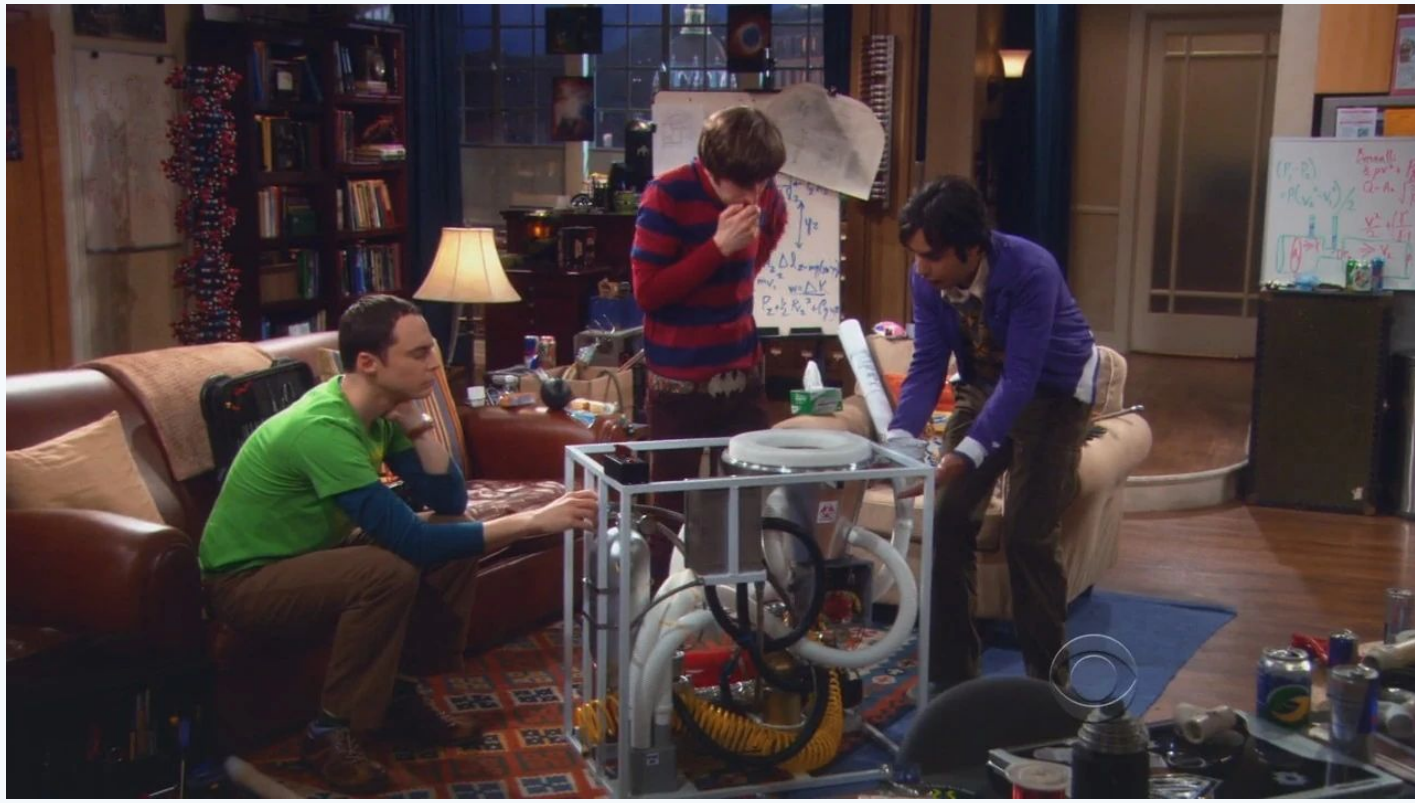
So why not just deliver incrementally?

- The reason is always the same: Having to synchronize the old system with the new system…bi-directionally
- Complicated, expensive, and generally awful…many companies give up

The Ask

CleanSlate TECHNOLOGY GROUP

- Replace 25-year-old ASP.NET Framework 4.6.1 app and AWS ECS services, legacy AWS account
- New serverless application stack in AWS
- Move ~5 Terabytes of SQL Server to a dozen DynamoDB tables
- One source of truth
- Both systems used simultaneously
- Clean cutover when ready to retire legacy



We've never built a space toilet, but we imagine it was a fun as this project

byrider

The Challenge

Strategies attempted and obstacles faced

CleanSlate
TECHNOLOGY GROUP

Strategy 1: DMS

- Views → DMS → Staging DynamoDB Table → Streams → Target DynamoDB Table
- "It worked for Oracle, so we're good!"
- Problem: can't use SQL Server Views with DMS
- Dead end 💀

Our DBA discovering the DMS didn't support SQL Server views a.k.a. "read the documentation" wasn't as dramatic as this "Campaign for North Africa" moment, but sure felt like it

Strategy 2: Roll-Your-Own

CleanSlate TECHNOLOGY GROUP

- Stored procedure model
- Unexpected benefits
  - Input parameters precisely control data volume per Lambda
  - Solved the "data purge" that never actually happened
- Not really a major shift, we were going to have to create views anyway

Scale

CleanSlate TECHNOLOGY GROUP

- Initial Load: 10s of millions of records per table
- Ongoing Load: 1000s of records per week
- Load impact at several points
  - SQL Server (still-running production DB)
  - DynamoDB Streams
  - Concurrent Lambda Invocations
  - SQS
  - CloudWatch

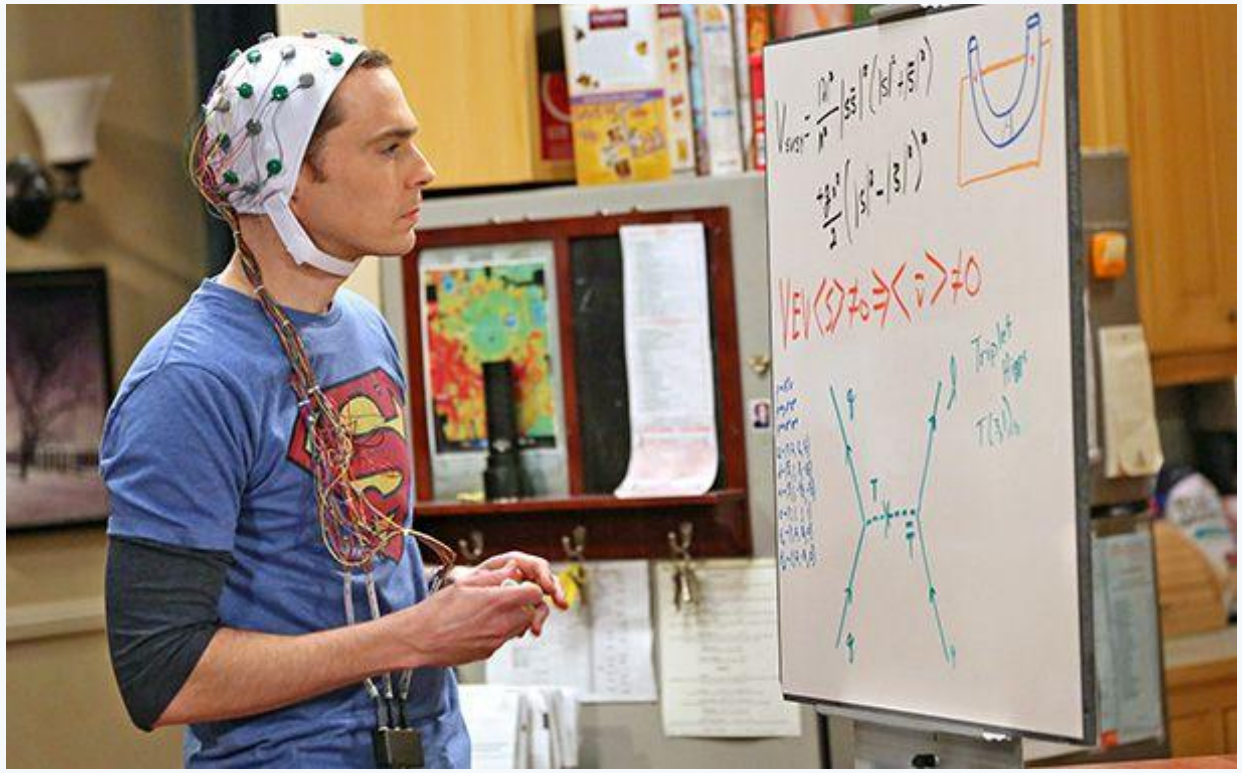
byrider

Daniel Schnipke, opening the floodgates on a full-size dataset

The Challenges

CleanSlate TECHNOLOGY GROUP

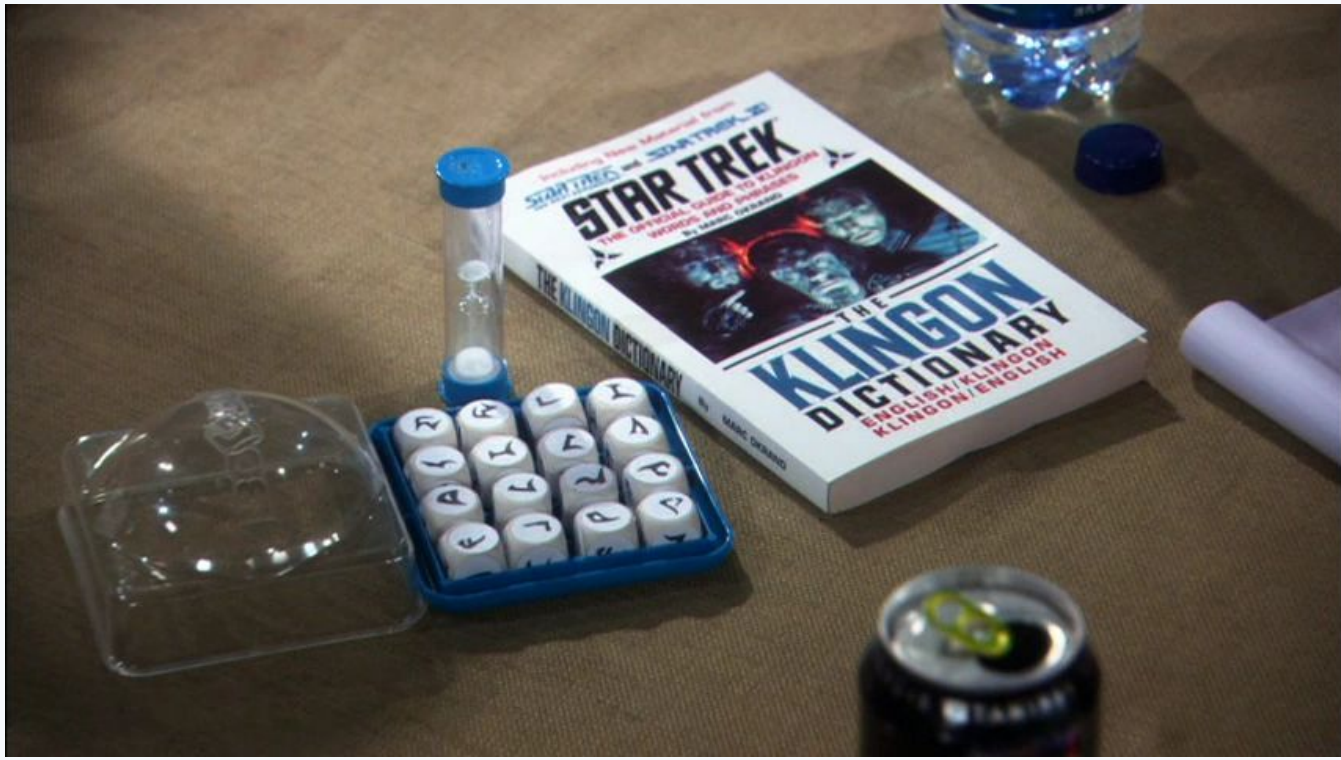
- Wildly different data implementations
- New system "fixed" problems at data layer
- Sync “fixed” ↔ “broken” data bidirectionally
- Circular updates / circuit breaker needed



Aaron Thomas vs the legacy system

Latency and User Experience

- Must be fast enough to avoid user confusion
- No thrashing back and forth during sync
- Example: user writes, edits, shouldn't see unmodified version



An example of high latency and questionable user experience: Klingon Boggle

The Key Problem

CleanSlate TECHNOLOGY GROUP

- Legacy SQL Server: sequential numbers (sometimes no IDs at all)
- New DynamoDB: ULID
- Brand new IDs, no place in legacy system
- Need bidirectional key mapping

When you have key problems

byrider

The Solution

Bidirectional sync architecture

All you need is a good flowchart

CleanSlate TECHNOLOGY GROUP

Solution: New to Legacy

CleanSlate TECHNOLOGY GROUP

- Cross-account DynamoDB stream
- Filter: `ConsistencyState == Pending`
- Lambda to Legacy SQL via stored procedures
- Near real-time sync
- Send data to the backend system, wait for it to come back

When you shoot something into the void…and get a response

Solution: Legacy to New

CleanSlate TECHNOLOGY GROUP

- Cross-account DynamoDB access
- EventBridge cron every 5 minutes
- SQL Row Version for predictable ordering
  - Store version number in Databridge DynamoDB
  - Fallback: date fields, then IDs (not great)
- Large batches: complete initial load + low latency
- `ConsistencyState = Current`

Dealing with Keys

CleanSlate TECHNOLOGY GROUP

- Record from old system first: Generate ULID, keep old ID on record
- Write to Databridge DynamoDB (pk = legacy ID)
- Record from new system first
  - Check for existing legacy ID
  - Write via stored procedure, get legacy ID back
  - Prevents duplicates in both directions

byrider

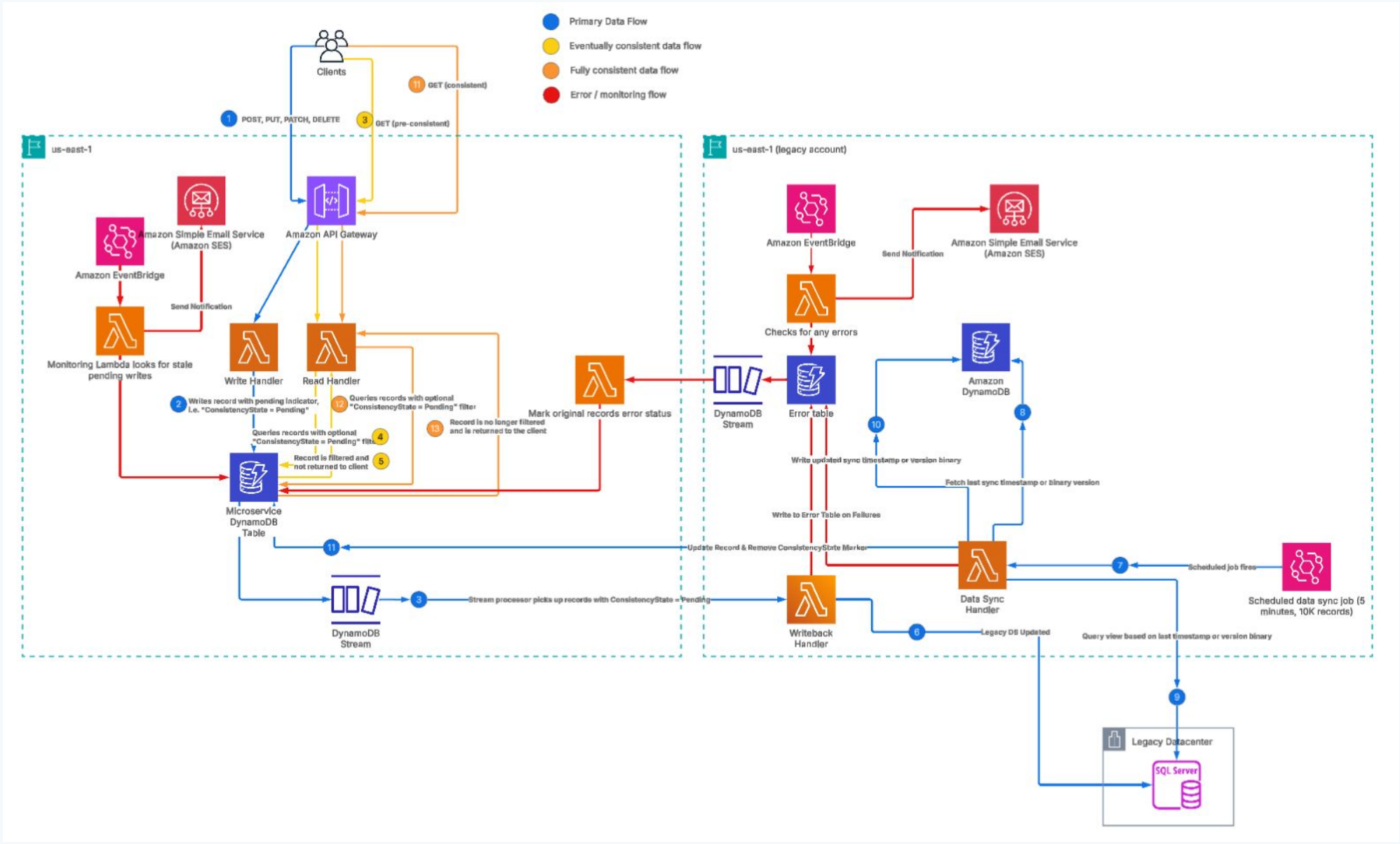
Front-End Strategy

CleanSlate TECHNOLOGY GROUP

- Borrowed from AWS eventual consistency patterns
- Give UI everything mid-cycle
- Let UX decide what to do with `ConsistencyState`
- Users stay productive during sync

If you keep knocking, “eventually” someone will probably open the door

Architecture Diagram



Questions?

CleanSlate
TECHNOLOGY GROUP

Continue the Conversation!

- Presentation downloads
- Speaker information
- Additional resources
- Office Hours registration and details
- Follow-up contact information

https://www.cleanslatetg.com/aws-community-day-resources/

CleanSlate TECHNOLOGY GROUP