WikiStream

Real-Time Wikipedia Edit Analytics Pipeline

500+	300+	≤3min	≤35min
EDITS/MINUTE	LANGUAGES	BRONZE SLA	E2E SLA

Data Engineering Portfolio Project

AWS Cloud • Medallion Architecture • Apache Iceberg 1.10

github.com/mdshihabullah/wikistream-event-data-pipeline

Shihab Ullah
Data Engineer

The Business Challenge

📊 Why Wikipedia Analytics?

Wikimedia is one of the largest collaborative platforms globally with **500-700 edits/minute** across 300+ language editions.

Metric	Value
Wikipedia Editions	300+ languages
Monthly Active Editors	280,000+
Bot Edits	20-30% of total
Peak Edit Rate	1,500+ edits/min

> **Data Source:** Wikimedia EventStreams (SSE)
> `stream.wikimedia.org/v2/stream/recentchange`

🎯 Target Use Cases

- ✓ **Content Monitoring** - Track edit velocity, trending pages
- ✓ **Vandalism Detection** - Rapid edits, large deletions
- ✓ **Regional Analysis** - Activity across language editions
- ✓ **Bot vs Human** - Monitor automated contributions
- ✓ **Risk Scoring** - Identify suspicious edit patterns

📍 Filtered Domains

High-activity: `en, de, ja, fr, zh, es, ru`

Regions: `asia_pacific, europe, americas, middle_east`

Shihab Ullah
Data Engineer

Architecture: System Context

System in context with external systems and users

System Responsibilities

- Ingest real-time SSE stream continuously (24/7)
- Buffer events in Kafka for reliability & replay
- Process through medallion layers (Bronze → Silver → Gold)
- Validate data quality at each transformation
- Store in ACID-compliant Iceberg tables

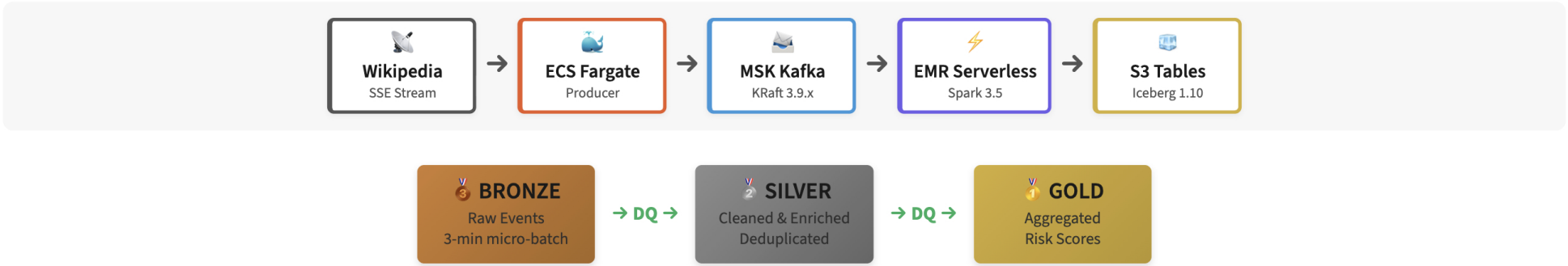
Key Outputs

- **Hourly Stats** — Volume, users, content by domain
- **Risk Scores** — User-level risk (0-100)
- **Daily Summary** — Platform health KPIs
- **DQ Audit Trail** — Evidence for compliance

Shihab Ullah
Data Engineer

Architecture: High-Level Flow

End-to-end data flow with medallion architecture and DQ gates



Supporting Services

Orchestration	Step Functions (self-looping)
Initial Trigger	EventBridge Scheduler
Auto-Recovery	Lambda + CloudWatch Alarm
Alerts	SNS (email notifications)

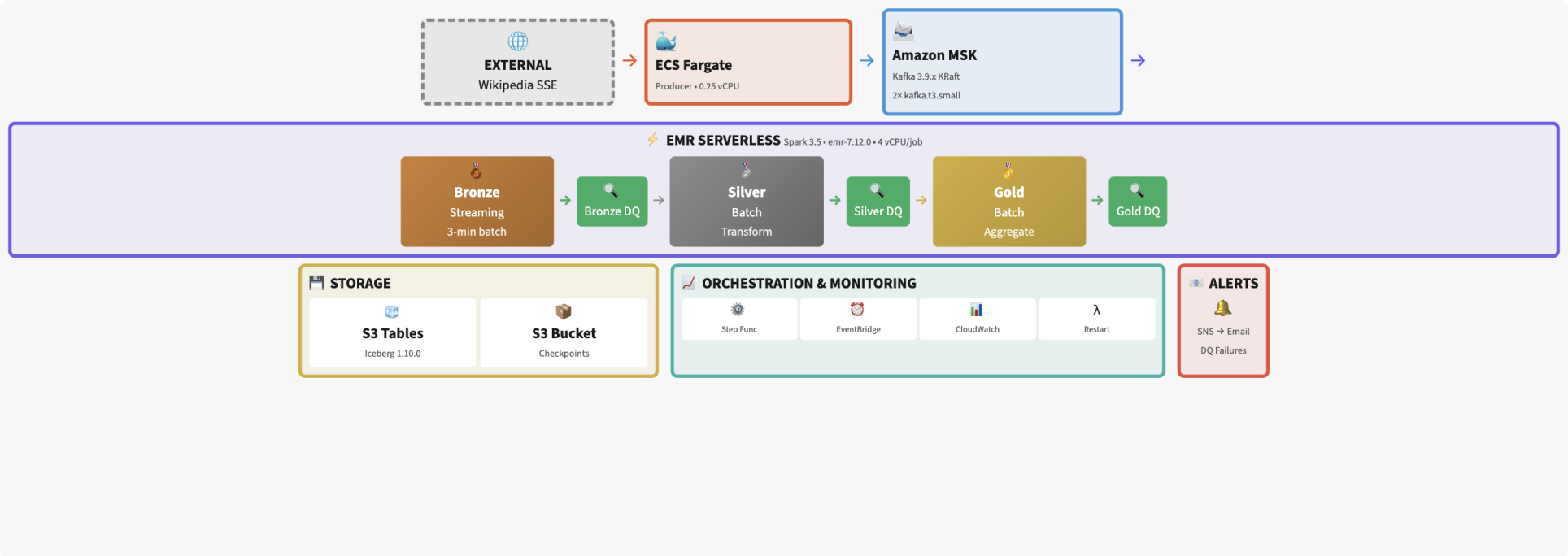
Data Quality Framework

Engine	AWS Deequ 2.0.7
Pattern	Gates block downstream
Audit	`dq_audit.quality_results`
On Failure	SNS Alert → Pipeline Stops

Shihab Ullah
Data Engineer

Architecture: Container Detail

AWS VPC (us-east-1)

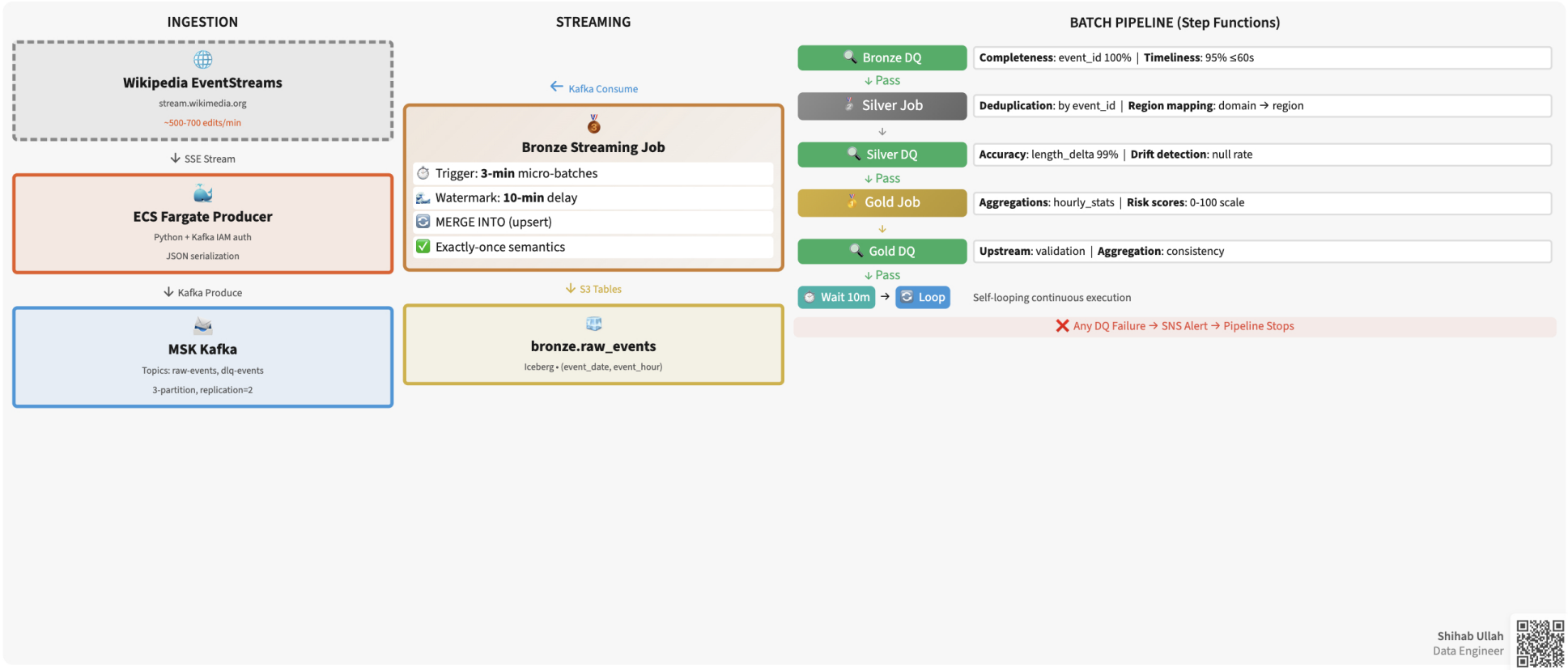


- ✓ **VPC:** 2 Availability Zones with private subnets
- ✓ **NAT:** Single NAT Gateway (cost-optimized)
- ✓ **Security:** IAM auth throughout all services

Shihab Ullah
Data Engineer

Data Pipeline Flow

End-to-end data journey from Wikipedia to Gold analytics — continuous processing



Processing: EMR Serverless Deep Dive

Spark 3.5 on emr-7.12.0 — Iceberg 1.10.0 integration with S3 Tables

🥉 Bronze Streaming

`bronze_streaming_job.py`

Type	Structured Streaming
Trigger	3-min micro-batches
Watermark	10 min (late data)
Write	MERGE INTO (upsert)
Resources	4 vCPU continuous

Key Operations:
- Deterministic event_id
- Kafka offset tracking
- Checkpoint management

🥈 Silver Batch

`silver_batch_job.py`

Type	Batch (Step Func)
Timeout	900s (15 min)
Input	bronze.raw_events
Output	silver.cleaned_events
Resources	4 vCPU on-demand

Transformations:
- Deduplication by event_id
- Region mapping
- Anonymity detection

🥇 Gold Batch

`gold_batch_job.py`

Type	Batch (Step Func)
Timeout	900s (15 min)
Input	silver.cleaned_events
Output	3 Gold tables
Resources	4 vCPU on-demand

Aggregations:
- hourly_stats by domain
- risk_scores by user
- daily_analytics_summary

Iceberg Configuration

Format	Version 2 (merge-on-read)
Compression	ZSTD
File Size	512 MB (compaction target)
Snapshots	48 hours retention

⚡ EMR Serverless Config

Release	emr-7.12.0 (Spark 3.5)
Quota	16 vCPU total
Max Concurrent	8 vCPU (Bronze + batch)
Auto-stop	5 min idle timeout

Shihab Ullah
Data Engineer

Data Quality Gates

AWS Deequ 2.0.7 — Gates block downstream processing on failure

Gate	Check	Rule	Threshold	Block
Bronze	Completeness	`event_id, domain, event_timestamp` non-null	100%	✓
	Completeness	`title, user, wiki` non-null	≥95%	⚠
	Timeliness	95th percentile event latency	≤60 sec	✓
	Validity	`event_type` in allowed set, `namespace ≥ 0`	100%	✓
Silver	Accuracy	`length_delta = length_new − length_old`	99%	✓
	Consistency	`is_valid = true` for all Silver records	100%	✓
Gold	Upstream	Bronze DQ + Silver DQ gates must have passed	Pass	✓
	Validity	`bot_percentage`, `risk_score` in 0-100 range	100%	✓

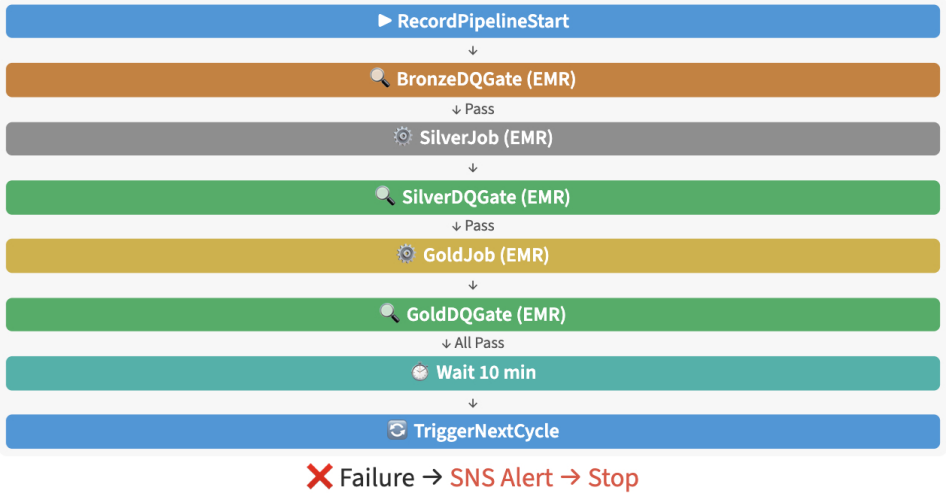
- ✓ **Audit Trail:** All DQ results logged to `dq_audit.quality_results`
- ✓ **Evidence:** Full JSON evidence stored for each check
- ✓ **Trend Analysis:** Historical data enables drift detection

Shihab Ullah
Data Engineer

Step Functions: Batch Pipeline

Orchestrates batch processing with DQ gates comprising Medallion architecture — **self-loops** every 10 minutes after successful completion

State Machine Flow



Pipeline Timing

Phase	Duration
Initial wait (Bronze data)	15 min
EMR cold start (JARs)	~5-8 min
DQ/batch per job	~2-3 min
Full pipeline	~15-25 min
Wait between cycles	10 min
Total cycle	**~25-35 min**

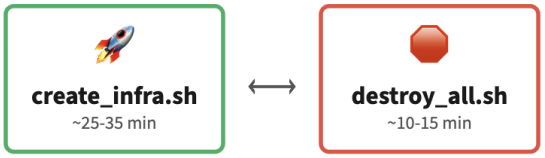
Resource (16 vCPU quota)

Bronze Streaming	4 vCPU (continuous)
Batch Jobs (each)	4 vCPU (sequential)
Max concurrent	8 vCPU

Shihab Ullah
Data Engineer

Infrastructure Lifecycle

Terraform IaC with idempotent create/destroy scripts for ease of implementation during development - **saving AWS bill** from running resources continuously



Create Infrastructure

```
./scripts/create_infra.sh
```

1. **Stop EMR if running** (for Terraform updates)
2. **Terraform apply** — VPC, MSK, EMR, ECS, S3 Tables
3. **Configure S3 Tables** — Intelligent-Tiering
4. **Build & push Docker** — Producer image to ECR
5. **Upload Spark jobs** — Including DQ module (dq.zip)
6. **Start ECS producer** — desired_count = 1
7. **Start Bronze streaming** — EMR Serverless job
8. **Schedule batch pipeline** — EventBridge (15 min)

Destroy Infrastructure

```
./scripts/destroy_all.sh
```

13-step teardown:

- Cancel running EMR jobs
- Stop EMR application
- Stop ECS service
- Delete EventBridge schedules
- Empty S3 buckets
- Delete S3 Tables (async)
- Terraform destroy
- Final verification

> **Idempotent:** Safe to run multiple times

Data Model: Medallion Layers

S3 Tables with Apache Iceberg 1.10.0 — Format Version 2

🥉 Bronze: Raw

Table: `bronze.raw_events`
Partition: `(event_date, event_hour)`

Key Fields:
- event_id (PK, deterministic)
- kafka_* metadata
- domain, title, user
- length_old/new/delta
- is_bot, comment

🥈 Silver: Cleaned

Table: `silver.cleaned_events`
Partition: `(event_date, region)`

Key Fields:
- event_id (PK)
- region (derived from domain)
- language (derived)
- is_anonymous (IP pattern)
- is_large_deletion/addition

🥇 Gold: Aggregated

Tables:
- `gold.hourly_stats`
- `gold.risk_scores`
- `gold.daily_analytics_summary`

Key Metrics:
- total_events, unique_users
- bot_percentage, risk_score

🔍 DQ Audit Tables

dq_audit.quality_results
Records all DQ gate check results
`run_id, layer, check_name, status, metric_value, evidence`

dq_audit.profile_metrics
Data profiling for drift detection
`column_name, null_rate, distinct_count, mean_value, percentiles`

S3 Tables Config

Format	2 (merge-on-read)
Compression	ZSTD
Compaction	512 MB
Snapshot	48 hours

Risk Scoring

High velocity	>50/hr	+40
Large deletions	>3	+30
Anonymous	>50%	+20
Cross-domain	>5	+10

HIGH (70+) • MEDIUM (40-69) • LOW (0-39)

Shihab Ullah
Data Engineer

Technology Decisions

Component	Choice	Rationale	Alternative
Queue	MSK (Kafka 3.9.x + KRaft)	No Zookeeper • IAM auth • DLQ support	MSK Serverless, Kinesis
Producer	ECS Fargate (0.25 vCPU)	Long-running SSE • Auto-restart • Pay-per-second	Lambda (15-min timeout)
Processing	EMR Serverless (Spark 3.5)	Zero idle cost • MERGE upserts • emr-7.12.0	EMR on EC2 (always-on)
Storage	S3 Tables (Iceberg 1.10.0)	ACID • Time-travel • Auto-compaction	Plain S3 + Parquet
Orchestration	Step Functions (self-loop)	90% cheaper than MWAA • Serverless	MWAA/Airflow ($250+/mo)
Data Quality	Deequ 2.0.7 + DQ Gates	Blocks downstream on failure • Audit trail	Great Expectations
IaC	Terraform 1.6+	Full AWS coverage • State management (S3, DynamoDB)	CloudFormation

- ✓ **Serverless-first:** Zero idle cost for batch processing
- ✓ **Pay-per-use:** Only pay during job execution
- ✓ **Fail-fast:** DQ gates block downstream on failure
- ✓ **Self-healing:** Lambda auto-restart on health check failure
- ✓ **Audit trail:** Evidence-rich DQ logging for compliance

Shihab Ullah
Data Engineer

Outcomes & Deliverables

✅ Implementation Complete

- ✓ **Real-time Ingestion:** Kafka + DLQ for malformed
- ✓ **Medallion Architecture:** Bronze/Silver/Gold + Audit
- ✓ **Idempotent:** MERGE INTO with deterministic ID
- ✓ **DQ Gates:** Block downstream on failure
- ✓ **Self-Looping Pipeline:** 10-min cycle intervals
- ✓ **Auto-Recovery:** Lambda restarts Bronze
- ✓ **Observability:** CloudWatch + SNS alerts
- ✓ **IaC:** Full Terraform (~2600 lines)

📊 Key Metrics

≤3min	≤35min	95%+	~$260
BRONZE SLA	E2E SLA	DQ PASS	MONTHLY

🛠️ Skills Demonstrated

MSK Kafka KRaft 3.9	**Iceberg** 1.10.0	**Spark** 3.5
EMR 7.12.0	**Step Func** Self-loop	**Terraform** 1.6+
Deequ 2.0.7	**Python** 3.12	**DQ Gates** Fail-fast

Shihab Ullah
Data Engineer