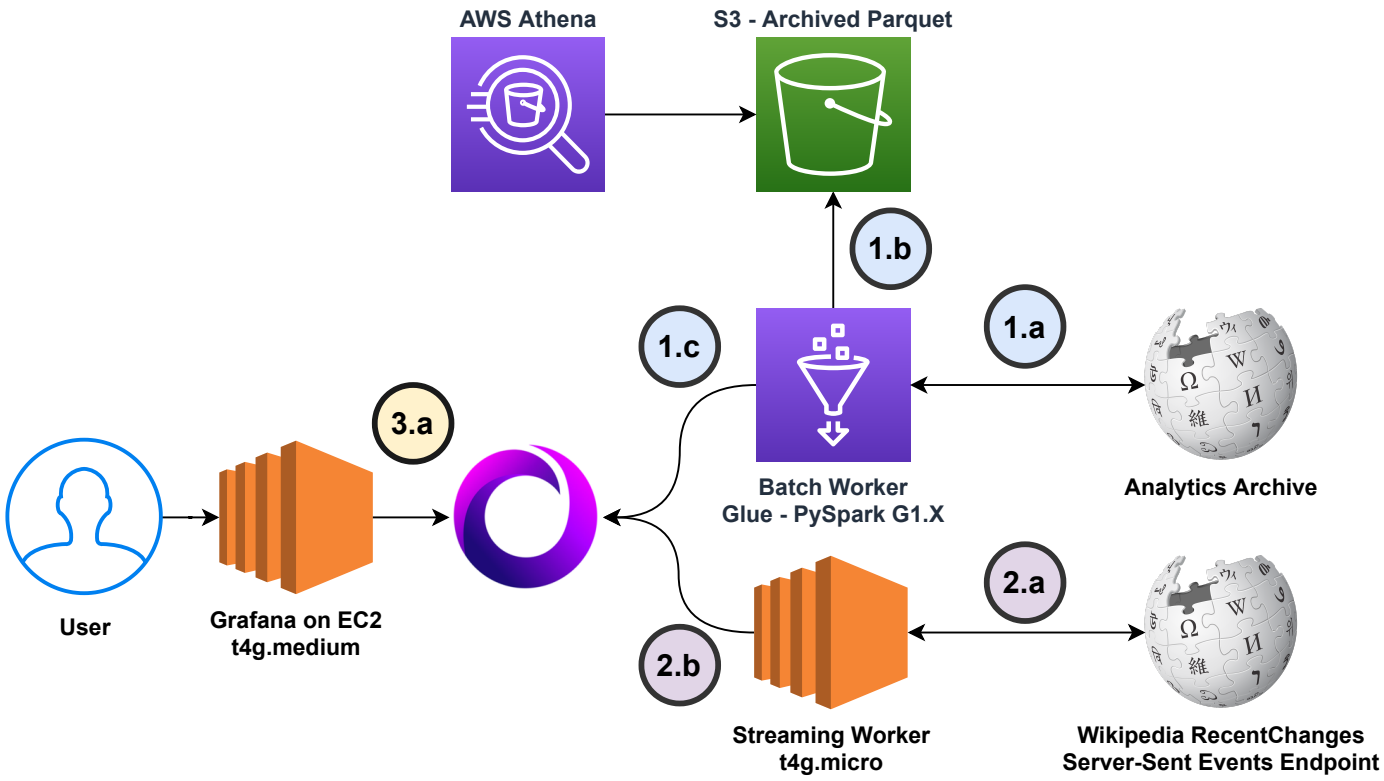
# Fig 1.1 - WikiMedia Analytics Architecture




**1.a** **BatchWorker** runs as a PySpark job on AWS Glue and reads from Wikipedia's MediaRequests and PageCounts.
(MediaCounts: ~30M uniques/day, PageCounts: ~140M pages/day)

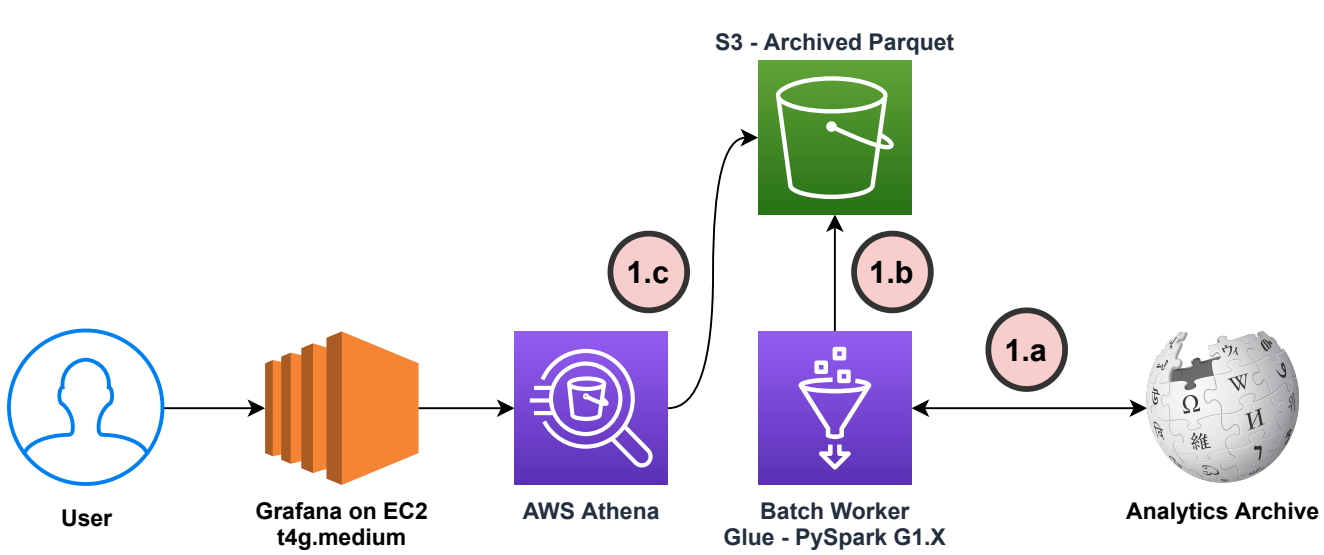
**1.b** **(optional) BatchWorker** writes the MediaRequests and PageCounts data as partitioned *.parquet files to AWS S3, the target file location has been configured as an Athena table for ad-hoc queries.

**1.c** **BatchWorker** writes the MediaRequests and PageCounts data to SingleStoreDB.

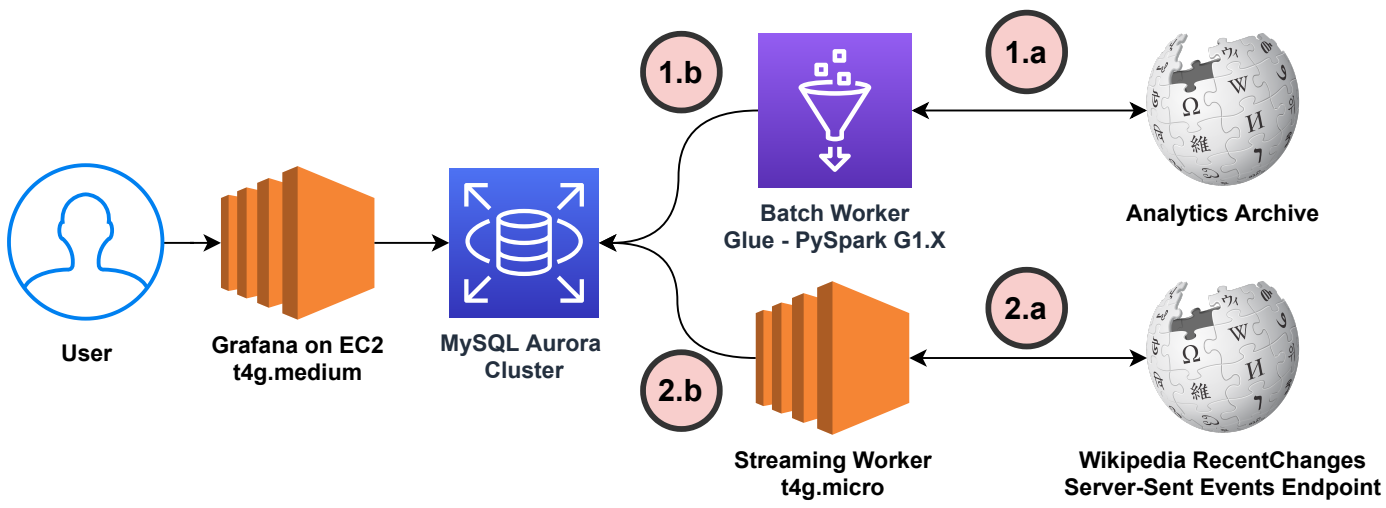
**2.a** **2.b** **StreamingWorker** is deployed to an EC2 instance and listens to the Wikipedia RecentChanges SSE endpoint, pushing events to SingleStoreDB on arrival. (~25 Edits/Second)

**3.a** Grafana is deployed to an EC2 instance and configured with **only** SingleStoreDB as a datasource for metrics created from streaming and batch jobs.

# Fig 1.2 - Alternative (Rejected) Architectures




**1.c** **BatchWorker** only writes to S3 / AWS Athena - with correct partitioning, this can be a solution for the batch datasources. To include streaming would need another component to bundle parquet files (either Kafka, Kinesis, SparkStreaming, or a TSDB). All of which add undesired complexity.




**1.b** **2.b** Both workers write to MySQL (or another OLTP RDBMS). Because the streaming data produces ~2M events/day, this may be OK. The batch worker produces ~150-200M records per day, which makes querying these sources untennable without very delicate performance tuning or a huge DB budget.