aws

Architecture Diagrams

Replaying Parallel Requests to Break a Monolith

Replaying Parallel Requests to Break a Monolith: Architecture Diagrams

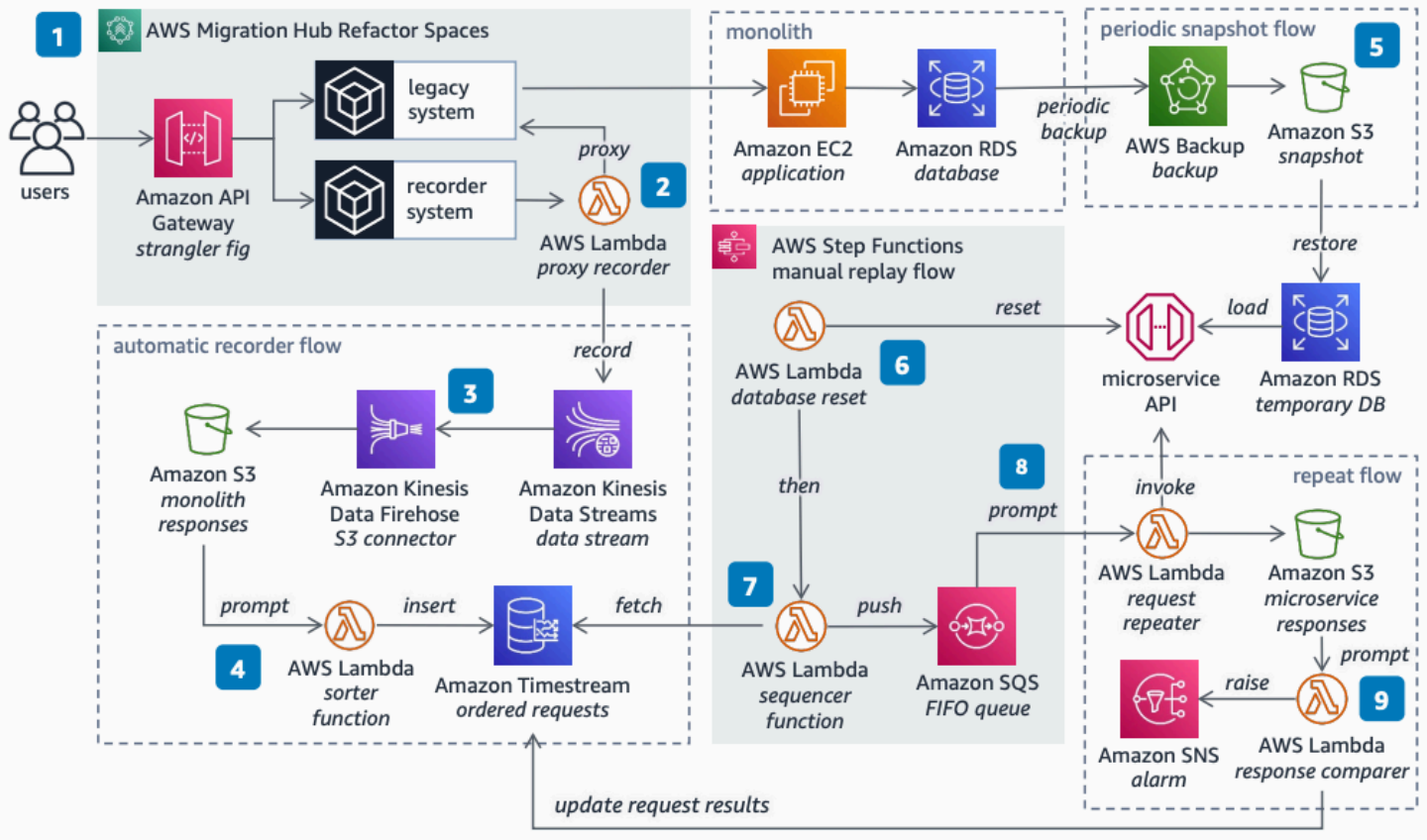
Table of Contents

Replaying Parallel Requests to Break a Monolith

Publication date: **June 24, 2022 (Diagram history)**

This architecture shows how to break your monolith with confidence by setting up a parallel run strategy combined with the strangler fig pattern. You proxy methods to be replaced with a microservice, store copies of user requests and monolith responses in a time-series database, then replay requests on your new microservice to compare responses.

Replaying Parallel Requests to Break a Monolith



The following steps describe the architecture:

1. Apply the strangler fig pattern with AWS Migration Hub Refactor Spaces to place an Amazon API Gateway API in front of your legacy cloud monolith on Amazon Elastic Compute Cloud and Amazon RDS. Re-route the endpoints-to-modernize into a recorder system that records all requests and responses.

2. Use a proxy recorder AWS Lambda function to proxy requests back to the legacy monolith, and push a copy of all request-and-response payloads into Amazon Kinesis Data Streams.
3. Use Amazon Kinesis Data Firehose to deliver the monolith-payload stream into an Amazon Simple Storage Service bucket.
4. Use a sorter function to store the payloads in Amazon Timestream in time-based order.
5. Periodically back up the monolith's database using AWS Backup to baseline the replay flow's start time.
6. Use an AWS Step Functions workflow to replay requests. First reset the microservice's temporary databases from a monolith database backup, then replay requests from the date and time of the backup.
7. Fetch all requests in the replay-time window sorted by date and time, then push them to a first-in-first-out (FIFO) queue using Amazon Simple Queue Service.
8. For each request in the queue, invoke the same request in the microservice's API and record the responses in an Amazon S3 bucket.
9. Compare responses from the same request sent to the monolith and microservice. Raise an alarm for any differences using Amazon Simple Notification Service, and store the final results in the requests database.

Further reading

For additional information, refer to the following resources:

- AWS Architecture Icons
- AWS Architecture Center
- AWS Well-Architected

Diagram history

To be notified about updates to this reference architecture diagram, subscribe to the RSS feed.

Change	Description	Date
Initial publication	Reference architecture diagram first published.	June 24, 2022

Note

To subscribe to RSS updates, you must have an RSS plugin enabled for the browser you are using.