1 Introduction

- 1 Introduction
  - 1 Imagine you run an e-commerce platform called ShopEase, hosted in the India (Mumbai) AWS region
  - 2 Your platform is performing well, and your Indian customers are happy with its fast and seamless experience
  - 3 Now, as your business grows, you're ready to expand globally to regions like the US and Europe to serve more customers and boost revenue
  - 4 However, expanding globally comes with technical challenges that could impact the experience of your new customers
- 2 Problem
  - 1 When customers from the US or Europe access your platform, your DynamoDB table in India must handle all their requests
  - 2 This creates several issues
    - 1 High Latency — Customers in the US or Europe experience slow responses because their requests have to travel to the India region and back
    - 2 Regional Failures — If the India region goes offline for maintenance or an unexpected outage, your platform becomes unavailable for all customers, including those in India, as the entire system depends on a single region
    - 3 Write Bottlenecks — With all global traffic hitting a single table in India, you may face performance issues during high demand, leading to delays in processing orderspic
- 3 Solution
  - 1 To overcome these challenges, you can use Amazon DynamoDB Global Tables
  - 2 Here's how it solves the problems
    - 1 Low Latency
      - 1 Global Tables replicate your DynamoDB data to specific regions that you choose, such as US-East (Virginia) and Europe (Ireland)
      - 2 Customers access the replica closest to them, ensuring fast performance
    - 2 High Availability — If one region (e.g., India) becomes unavailable, your application continues running in other regions without disruption
    - 3 Automatic Data Sync
      - 1 DynamoDB Global Tables automatically replicate and synchronize data across all regions.
      - 2 Whether an order is placed in the US, Europe, or India, all regions remain up-to-date in real time

2 How It Works

- 1 Table Creation in the Primary Region
  - 1 Start by creating a DynamoDB table in your primary AWS region (e.g., India (Mumbai))
  - 2 This acts as the source table for replication
- 2 Adding Replica Tables
  - 1 Specify the additional AWS regions where you want the table replicated (e.g., US-East (Virginia) and Europe (Ireland))
  - 2 DynamoDB automatically creates replicas in these regions.
- 3 Data Replication
  - 1 DynamoDB uses DynamoDB Streams to track item-level changes (inserts, updates, deletes) in any table
  - 2 Changes are sent to all replica tables in the background, and they update shortly after the original write is completed
- 4 Multi-Region Writes
  - 1 Each replica table is writable, allowing applications to write to the closest replica
  - 2 DynamoDB resolves conflicts using the last writer wins strategy, based on timestamps
- 5 Consistency Across Regions
  - 1 DynamoDB ensures eventual consistency for replicated data
  - 2 While writes are applied locally first, they propagate to all replicas shortly afterward
- 6 High Availability — If one region becomes unavailable, your application can automatically switch to another region's replica table for uninterrupted operations

3 Example Workflow

- 1 Imagine a ShopEase customer in the US places an order
- 2 The order is written to the US-East (Virginia) replica table
- 3 DynamoDB Streams detect the change and replicate it to the India and Europe tables
- 4 Customers in other regions (like India) can instantly see the updated inventory after replication completes
- 5 This workflow ensures all customers see the same consistent data, no matter where they are.