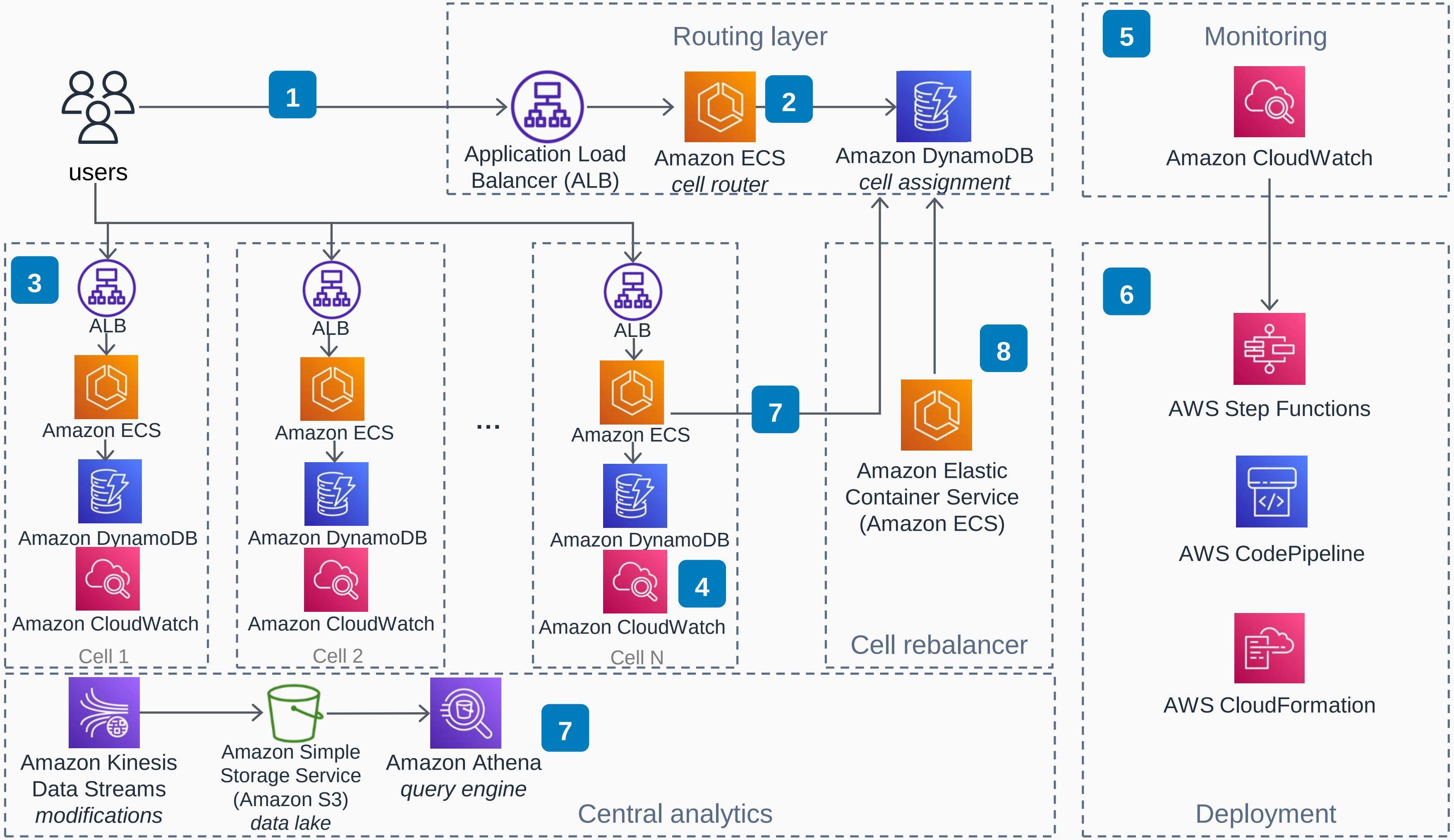
Guidance for Cell-Based Architecture on AWS

A way to shard an application as a whole

At a very large scale, a single load balancer (such as a VPC and other elements) become complex. Here, everything is redundant except for a thin routing layer. Metrics and analytics are aggregated, and deployments and changes are automated.



1. Clients connect to the routing layer. The routing layer redirects the client to the assigned cell using an HTTP redirect.
2. Routing information (user to cell mapping) is stored in **Amazon DynamoDB**. There is a fixed number of independent clusters that store copies of the data. For new users, the cell router pushes the new user information to all clusters.
3. The architecture is divided into a large number of independent cells of fixed size. The cells contain all application logic and storage.
4. Each cell has monitoring and alerting capabilities using **Amazon CloudWatch**.
5. There is also a central dashboard which contains aggregated information (such as number of cells with and without errors).
6. Cell creation and update is automated using **AWS Step Functions**, **AWS CodePipeline**, **AWS CodeDeploy**, and **AWS CloudFormation**. Updates are first deployed to a canary cell. Disaster recovery for cells is fully automated.
7. Changes are streamed from all cells to a central data lake, where they can be queried using SQL in **Amazon Athena**.
8. A rebalancer can move users between cells, and also create new cells as needed. After a successful move, it updates the user-to-cell assignment. The old cell retains a marker to redirect clients to the new cell (*not pictured in the diagram*).

Reviewed for technical accuracy December 1, 2023

AWS Reference Architecture