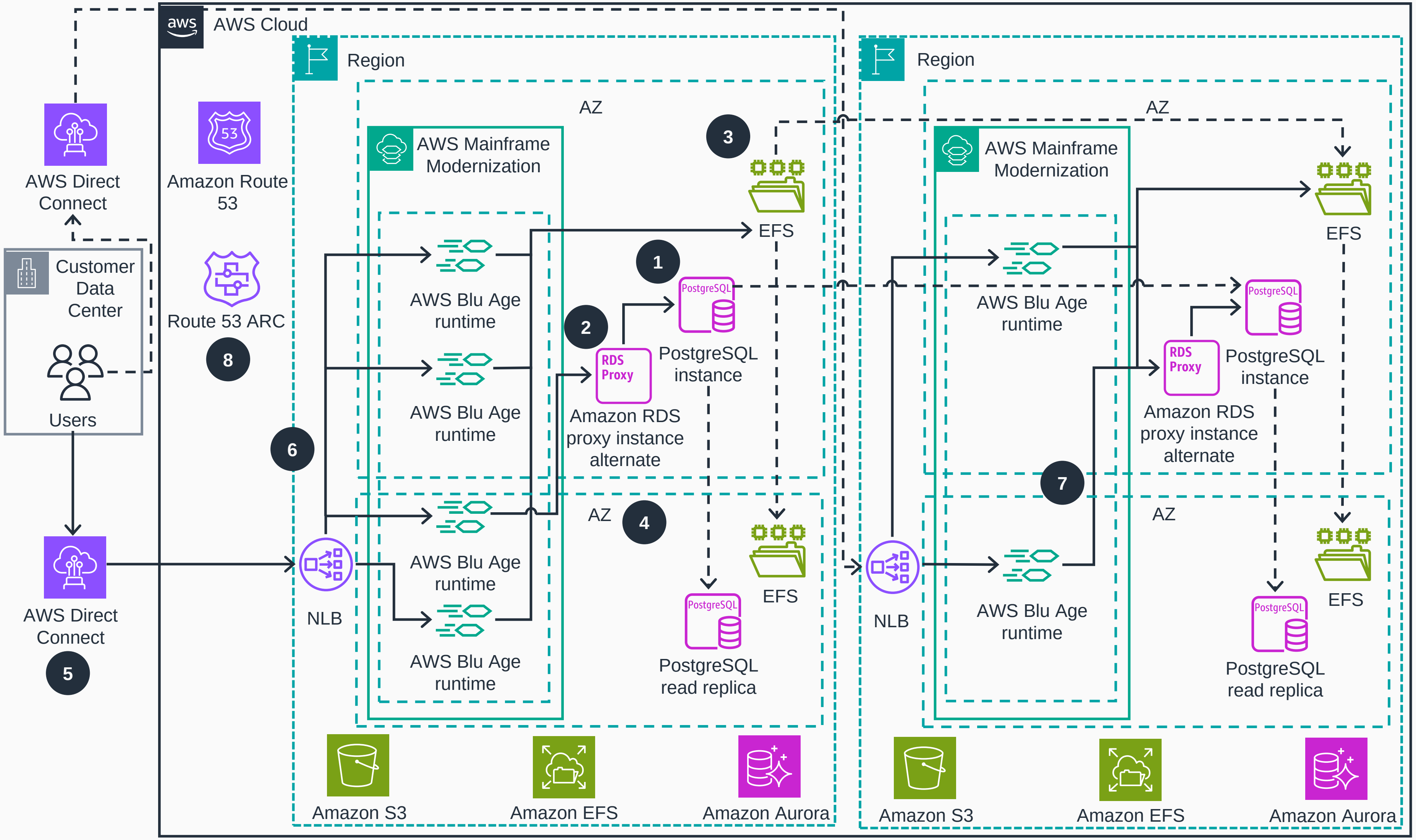
Guidance for Warm Standby Using AWS Mainframe Modernization Refactor with AWS Blu Age

This architecture diagram shows how to implement a warm standby disaster recovery environment for a refactored application.



1 Create a global **Amazon Aurora** Postgres database with a primary Region and secondary Region cluster.

2 Create **Amazon Relational Database Service (Amazon RDS)** proxies for the database to help with pooling and database connections.

3 Create an **Amazon Elastic File System (Amazon EFS)** file system. Turn on regional replication to create a replica in the secondary Region.

4 Create a multi-Availability Zone (AZ) **AWS Mainframe Modernization with AWS Blu Age** environment in the primary Region.

5 Configure **AWS Direct Connect** to establish a dedicated network connection between your on-premises infrastructure and AWS.

6 Create and deploy the **AWS Blu Age** application from **AWS Mainframe Modernization** using the **Amazon RDS** proxy and **Amazon EFS** mount target. **AWS Mainframe Modernization** automatically provisions a Network Load Balancer (NLB) to distribute traffic to the modernized application.

7 Repeat Steps 2-5 in the secondary Region. While creating an environment in secondary Region, choose the scaled-down version of configuration to ensure the application is up and on standby. During the disaster recovery invocation process, you can use the APIs provided by **AWS Mainframe Modernization** to scale up resources in the secondary Region.

8 **Additional Recommendations:** Steps 1-7 help you set up a warm standby architecture for your application in place. When creating a disaster recovery playbook, you can follow the additional best practices to automate the process:

- Use **Amazon Route 53**-hosted zones and **Route 53** Application Recovery Controller (ARC) to route the traffic between primary and secondary Regions.
- After subscribing to **Amazon RDS** events that are generated from global database failover, you can use **AWS Lambda** functions to scale the resources in the secondary Region.

Reviewed for technical accuracy July 15, 2024

AWS Reference Architecture