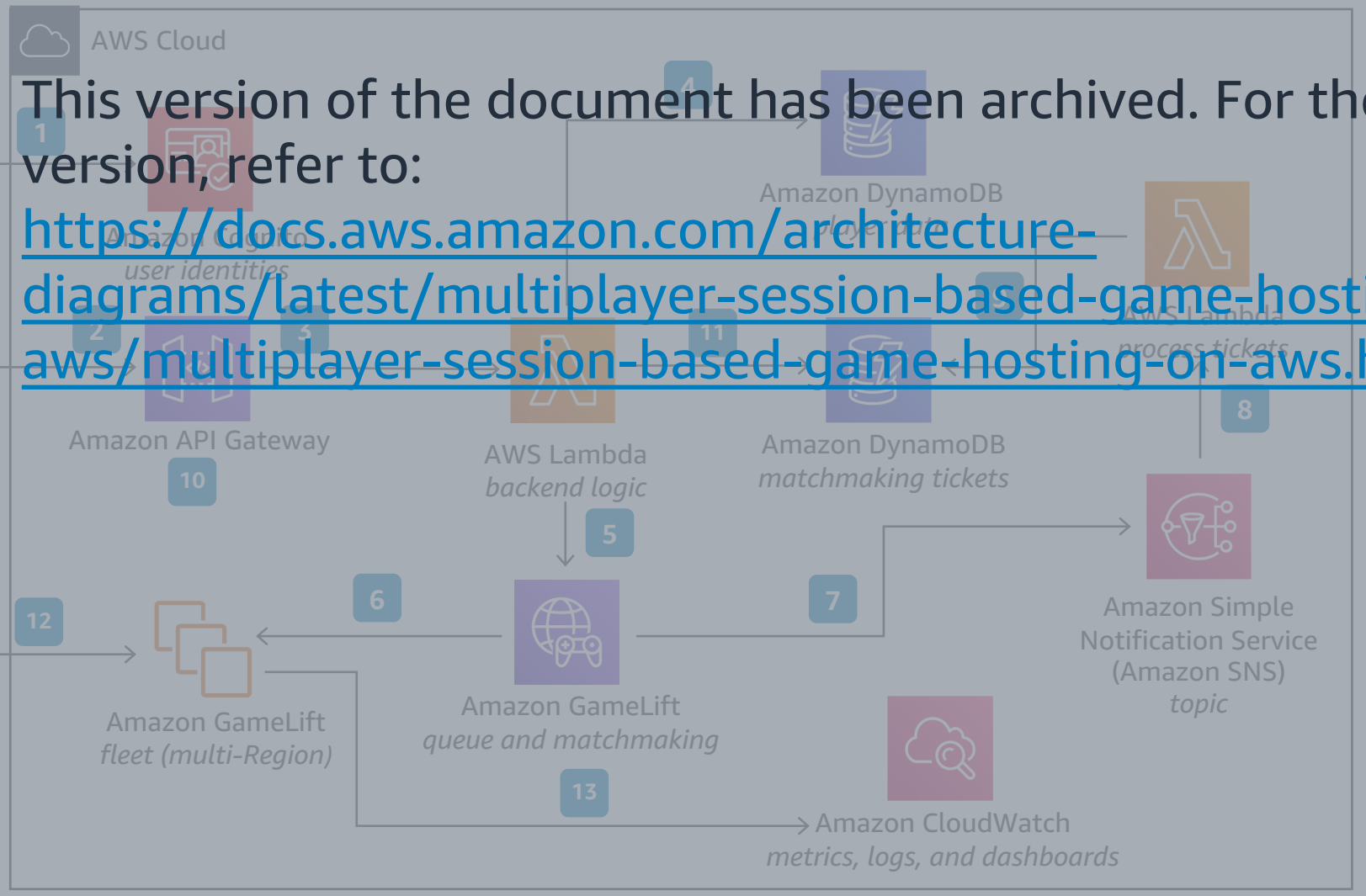
Multiplayer Session-based Game Hosting on AWS

Using Amazon GameLift multi-Region fleets and a serverless backend solution to host a session-based multiplayer game.



1. The game client requests an **Amazon Cognito** identity and temporary AWS credentials.
2. The client signs a matchmaking request to **API Gateway** with the temporary credentials. The request includes client latency information to supported AWS Regions.
3. **API Gateway** calls an **AWS Lambda** function with player identity information.
4. The **Lambda** function gets player skill level from **DynamoDB** table.
5. The **Lambda** function requests matchmaking from **GameLift FlexMatch** with player skill and latency data.
6. **GameLift FlexMatch** creates a match with multiple players, and a **GameLift** queue allocates a session in a **GameLift** fleet location based on the latency data.
7. **GameLift FlexMatch** publishes an event to **Amazon SNS** on matchmaking success.
8. **Amazon SNS** triggers a subscribed **Lambda** function for ticket processing.
9. The **Lambda** function stores the ticket result in a **DynamoDB** table.
10. The game client polls for matchmaking success on a defined interval from **API Gateway.**
11. The **Lambda** function checks matchmaking information from the **DynamoDB** table and informs the client of a successful match by returning server IP, port, and player session ID.
12. The client connects directly to the server and sends the player session ID. GameLift Server SDK is used to validate the player session.
13. Game servers send logs and metrics to **Amazon CloudWatch** with **CloudWatch** agent.

Reviewed for technical accuracy February 23, 2022

AWS Reference Architecture