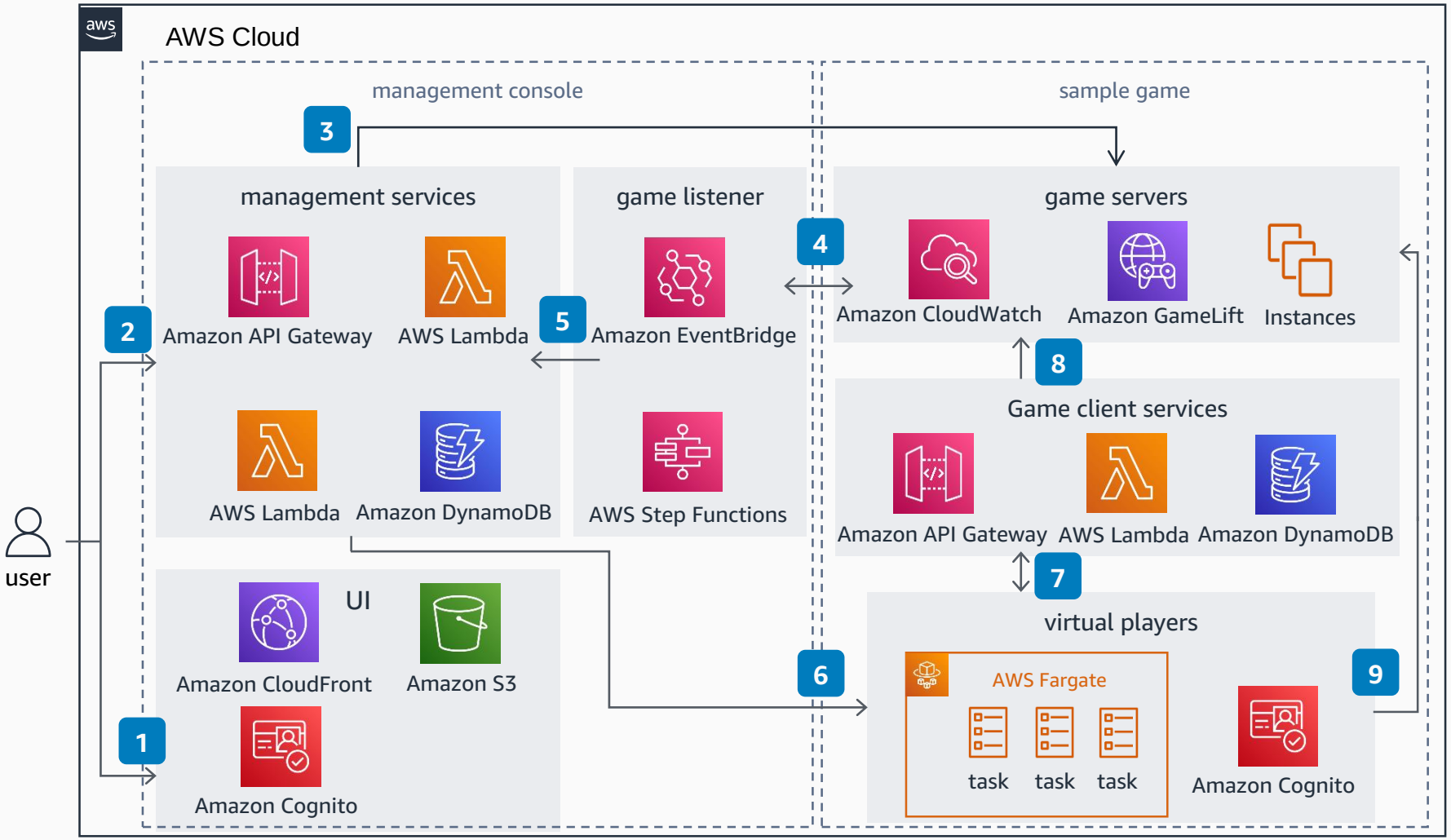
Guidance for GameLift Testing on AWS

The GameLift Testing Toolkit automatically detects GameLift deployments, and provides a near real-time view of player activity, letting users trial different configurations and quickly identify bugs and inefficiencies.



1. **Amazon CloudFront** provides access to the toolkit web console, which is stored on **Amazon Simple Storage Service** (Amazon S3). Toolkit console users authenticate with **Amazon Cognito** user pools.
2. The web console connects with an **Amazon API Gateway** WebSocket API, which calls an **AWS Lambda** function to process requests and store data in **Amazon DynamoDB**.
3. The **Lambda** function can query and update **Amazon GameLift** infrastructure directly.
4. The game listener receives **GameLift** events through **Amazon EventBridge** and uses **AWS Step Functions** to poll and receive infrastructure updates from **GameLift** and **Amazon CloudWatch.**
5. The game listener dispatches aggregated event data over a custom **EventBridge** event bus. The events are received by a **Lambda** function which stores data in **DynamoDB** and calls **API Gateway** to send data to connected web console users for display.
6. The web console user can also launch virtual player tasks on **AWS Fargate** to simulate real game clients.
7. Each virtual player task requests an Amazon **Cognito** guest identity and connects into the game client services **API Gateway** to request matchmaking.
8. The **Lambda** functions call **GameLift FlexMatch** to start the matchmaking process and place successful matches onto a fleet using game session queues.
9. The virtual player tasks connect with **GameLift** fleet instances to start their game sessions.

Reviewed for technical accuracy November 16, 2022

AWS Reference Architecture