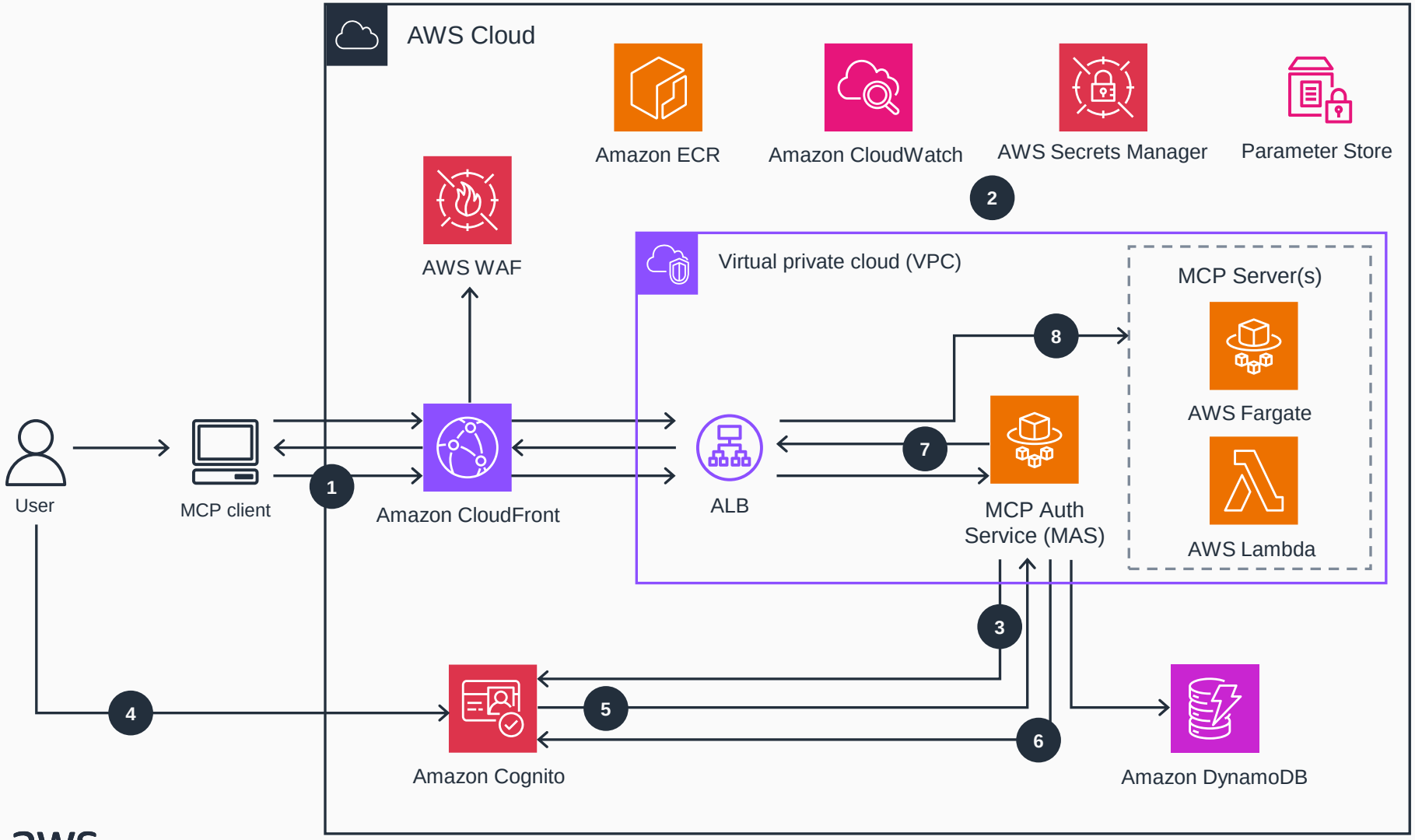
Guidance for Deploying Model Context Protocol Servers on AWS

Securing MCP servers with Amazon Cognito OAuth (using 2025-03-26 specification)

This architecture diagrams illustrates how to effectively support the secure deployment of MCP servers on AWS. It shows the key components and their interactions, providing an overview of the architecture's structure and functionality.



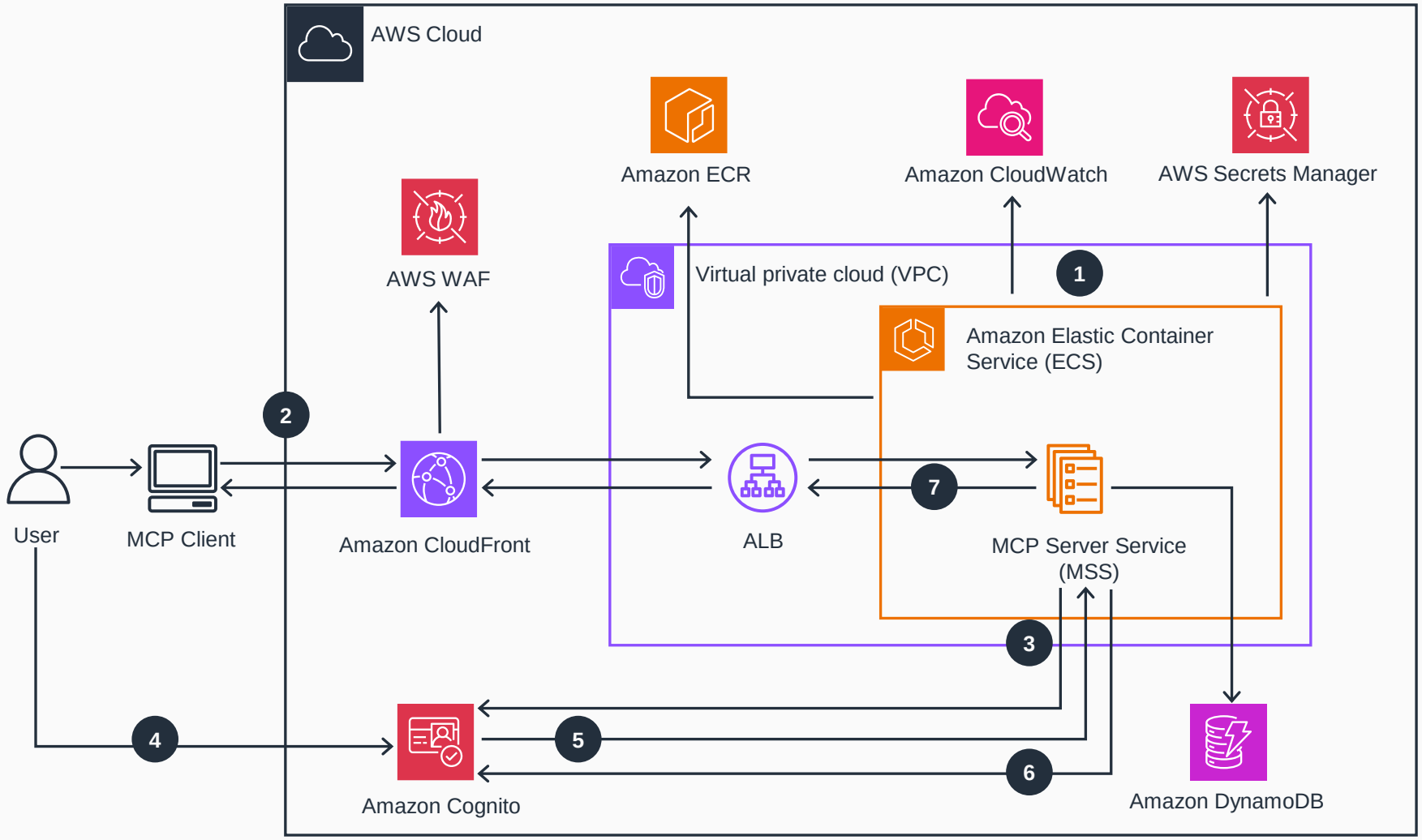
1. The MCP client initiates the OAuth request. The request passes through the **Amazon CloudFront** distribution (protected by **AWS WAF**) and Application Load Balancer (ALB) to the MCP Auth Service (MAS).
2. MAS runs in a secure virtual private cloud (VPC) on **AWS Fargate** using container images published to **Amazon Elastic Container Registry (Amazon ECR)**. The server writes task logs to **Amazon CloudWatch,** retrieves configurations from Parameter Store (a capability of **AWS Systems Manager**), and retrieves sensitive variables from **AWS Secrets Manager**.
3. The MAS redirects the user to **Amazon Cognito** for authentication. The server generates a unique session ID and stores the client information and OAuth parameters in **Amazon DynamoDB** with a 24-hour time-to-live (TTL).
4. The user authenticates directly with **Amazon Cognito** in the browser. **Amazon Cognito** validates credentials and issues authorization codes for successful authentications.
5. After successful authentication, **Amazon Cognito** redirects back to the MAS callback URL. The MAS retrieves the original session data from **DynamoDB** using the session ID passed as state parameter.
6. MAS exchanges an authorization code for tokens from **Amazon Cognito**. MAS stores a mapping between the MAS authorization code and **Amazon Cognito** tokens in **DynamoDB** with a 10-minute TTL.
7. MAS generates its own access token and sends to the MCP client. The response flows back through ALB and **CloudFront**. MAS generates refresh tokens stored in **DynamoDB** with a 30-day TTL for enabling token refresh without re-authentication.
8. The MCP client uses the received access token when making API requests to MCP servers on **Fargate** and **AWS Lambda**, which validate the token's signature and verify its status with **Amazon Cognito** before processing the request and returning a response to the client.

Reviewed for technical accuracy September 2, 2025

AWS Reference Architecture

Guidance for Deploying Model Context Protocol Servers on AWS

Securing MCP servers hosted on an ECS Cluster with Amazon Cognito OAuth (using 03-26-2025 specification)
This architecture diagrams illustrates how to effectively support the secure deployment of MCP servers on AWS. It shows the key components and their interactions, providing an overview of the architecture's structure and functionality.



1. MCP Server Service (MSS) runs in a secure Virtual Private Cloud (VPC) on **Amazon ECS** using container images published to **Amazon Elastic Container Registry (ECR)**. The server writes task logs to **Amazon CloudWatch** and retrieves sensitive variables from **AWS Secrets Manager**.
2. MCP Client initiates the OAuth request. The request passes through the **Amazon CloudFront** distribution (protected by **AWS WAF**) and Application Load Balancer (ALB) to the MSS.
3. The MSS redirects the user to **Amazon Cognito** for authentication. The server generates a unique session ID and stores the client information and OAuth parameters in **Amazon DynamoDB** with a 24-hour time-to-live (TTL).
4. The User authenticates directly with **Amazon Cognito** in the browser. **Amazon Cognito** validates credentials and issues authorization codes for successful authentications.
5. After successful authentication, **Amazon Cognito** redirects back to MSS callback URL. The MSS retrieves the original session data from **DynamoDB** using the session ID passed as state parameter.
6. MCP Server exchanges an authorization code for tokens from **Amazon Cognito**. MSS stores a mapping between the MSS authorization code and **Amazon Cognito** tokens in **DynamoDB** with a 10-minute TTL.
7. MSS generates its own access token and sends to MCP Client. Response flows back through ALB and **CloudFront**. MSS generates refresh tokens stored in **DynamoDB** with a 30-day TTL for enabling token refresh without re-authentication.

Reviewed for technical accuracy September 2, 2025

AWS Reference Architecture