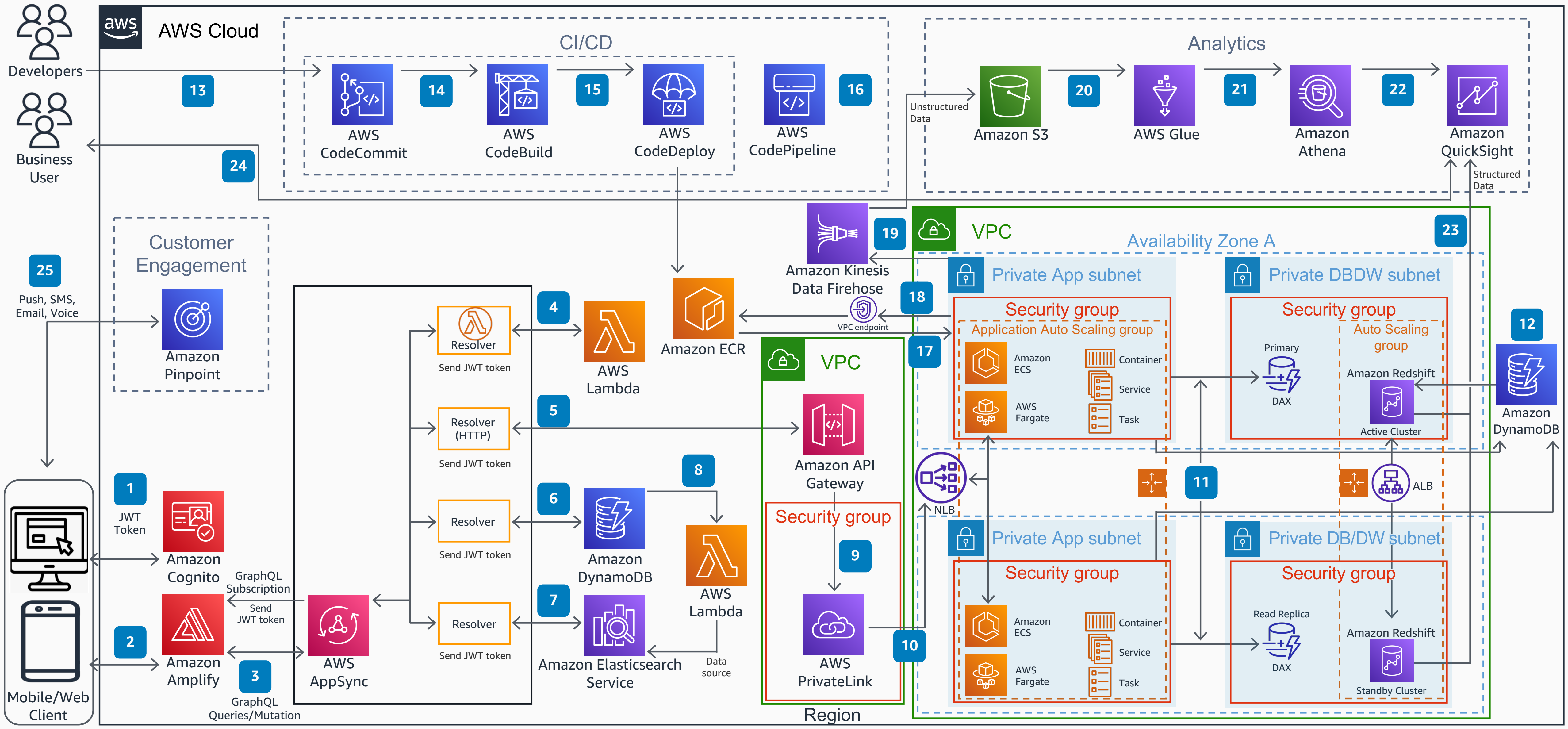
Modern Serverless Mobile/Web Application Architecture Integrated with CI/CD and Analytics Use Case

This diagram shows how to build a modern serverless mobile/web application in AWS for mobile/web clients, using AWS AppSync for frontend and ECS Fargate containers for backend application, along with Continuous Integration and Delivery and analytics to derive insight from application logs and structured data using the Amazon QuickSight dashboard.



Reviewed for technical accuracy May 10, 2021

AWS Reference Architecture

1 Users authenticate via **Amazon Cognito** user pools by retrieving the JWT token, and they then use those tokens to retrieve AWS credentials that allow their app to access other AWS services.

2 The web/mobile client interacts with **AWS Amplify** frameworks, which allow communication with backend services with iOS, Android, web, and React Native front ends.

3 The authenticated clients make API calls to **AWS AppSync** to perform GraphQL operations such as queries, mutations, and subscriptions.

4 The **AWS Lambda** resolvers communicate with Lambda with temporary **AWS Identity and Access Management (IAM)** credentials based on assumed IAM roles. A JWT token specific to the authenticated user is forwarded to **Lambda** for processing.

5 The HTTP resolver and the endpoints are protected with temporary IAM credentials based on assumed IAM roles. A JWT token specific to the authenticated user is forwarded to **Amazon API Gateway**.

6 The **Amazon DynamoDB** resolver enables connecting existing tables to a GraphQL schema by creating a data source to read, write, and subscribe to real-time data. The JWT token specific to the authenticated user is forwarded to Lambda in step 8.

7 The **AWS AppSync** resolver for **Amazon Elasticsearch Service (Amazon ES)** enables you to use GraphQL to store and retrieve data from existing **Amazon ES** domains, map an incoming GraphQL request into an **Amazon ES** request, and then map the **Amazon ES** response back to GraphQL.

8 **AWS Lambda** sends data to the **Amazon ES** domain from **DynamoDB** whenever new data arrives in the database table. This triggers an event notification to **Lambda**, which runs and performs the indexing.

9 **API Gateway** uses **AWS PrivateLink** to encapsulate connections between **API Gateway** and **Amazon ECS on AWS Fargate** configured in another **Amazon Virtual Private Cloud (VPC)** with a security group controlling access.

10 The **Network Load Balancer (NLB)** is configured with a specific port assigned to each service through private integrations towards the **Amazon ECS on AWS Fargate** cluster, running across multiple Availability Zones for high availability in two different private subnets, and configured with **Application Auto Scaling**.

11 The application workload hosted in **Amazon ECS on AWS Fargate** containers access the **Amazon DynamoDB Accelerator (DAX)** in the in-memory cache layer to retrieve frequently-accessed information, to improve the performance of the application and send the response faster.

12 **DynamoDB** offers the benefit of performance at scale with no server management (serverless). It is typically built for mission-critical workloads, including support for atomicity, consistency, isolation, durability (ACID) transactions for a broad set of applications that require complex business logic.

13 **AWS CodeCommit** is a fully managed service that acts as a repository for storing application code whenever the developer modifies or commits the code.

14 **AWS CodeBuild** is a continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy on a dynamically-created build server. After the build is successful, the pipeline moves to the deploy stage.

15 **AWS CodeDeploy** is a fully-managed deployment service that automates software deployments to **AWS Fargate**. The deployments contain the code or application (associated to new features to be deployed based on developer changes) running **CodeDeploy** agents.

16 **AWS CodePipeline** is used to create an end-to-end pipeline that fetches the application code from **CodeCommit**, builds and tests using **CodeBuild**, and finally deploys using **CodeDeploy** from **Amazon Elastic Container Registry** (**Amazon ECR**) to the **Fargate** serverless platform for handling user traffic.

17 The latest Docker images generated via the build process are stored in **Amazon ECR** and pulled for deployment in **Fargate** based on the developer changes to the source code (see step 13) in **Fargate**, and they run across multiple Availability Zones for high availability (HA) with the latest version.

18 **Amazon ECS on AWS Fargate** connects to **Amazon ECR** via **Amazon VPC** by configuring **ECR** to use an interface **Amazon VPC** endpoint. **Amazon VPC** interface endpoints enable private access to **ECR** APIs through private IP addresses. **AWS PrivateLink** restricts all network traffic between the **Amazon VPC** and **ECR** using the Amazon network.

19 The unstructured data from the **Amazon ECS on AWS Fargate** cluster is sent to **Amazon Kinesis Data Firehose** in near real-time towards the data lake in **Amazon Simple Storage Service** (**Amazon S3**). **Kinesis Data Firehose** is serverless, requires no administration, and has pay-as-you-go pricing. You pay only for the volume of data you transmit and process through the service.

20 The **AWS Glue** extract, transform, load (ETL) connects to data stored in **Amazon S3** using the **AWS Glue Data Catalog** to store metadata such as table and column names. The **AWS Glue** crawler retrieves the information automatically; alternatively, you can manually add a table and enter the schema information yourself.

21 **Amazon Athena** is an interactive query service that makes it easy to analyze data registered with the **AWS Glue Data Catalog**. **Athena** uses Presto to process data manipulation language (DML) statements to process the data definition language (DDL) statements that create and modify the schema.

22 **Amazon QuickSight** can create and publish interactive business intelligence (BI) dashboards, and, using **Athena** as your data source, select the database and tables to analyze and start visualizing.

23 **Amazon Redshift** is a cloud data warehouse service for analytics. **Amazon Redshift** complements **Amazon DynamoDB** with advanced business intelligence capabilities and a powerful, SQL-based interface. **DynamoDB** table data is copied into **Amazon Redshift**, which can perform complex data analysis queries on that data, including joins with other tables in your **Amazon Redshift** cluster.

24 User management is different between the **QuickSight** Standard and Enterprise editions (see Different Editions of Amazon QuickSight). However, both editions support identity federation, or Federated Single Sign-On (SSO), through Security Assertion Markup Language 2.0 (SAML 2.0).

25 **Amazon Pinpoint** is used for all marketing communication scenarios. It segments the campaign audience to reach the right customers, and personalizes messages with the right content.

Reviewed for technical accuracy May 10, 2021

AWS Reference Architecture