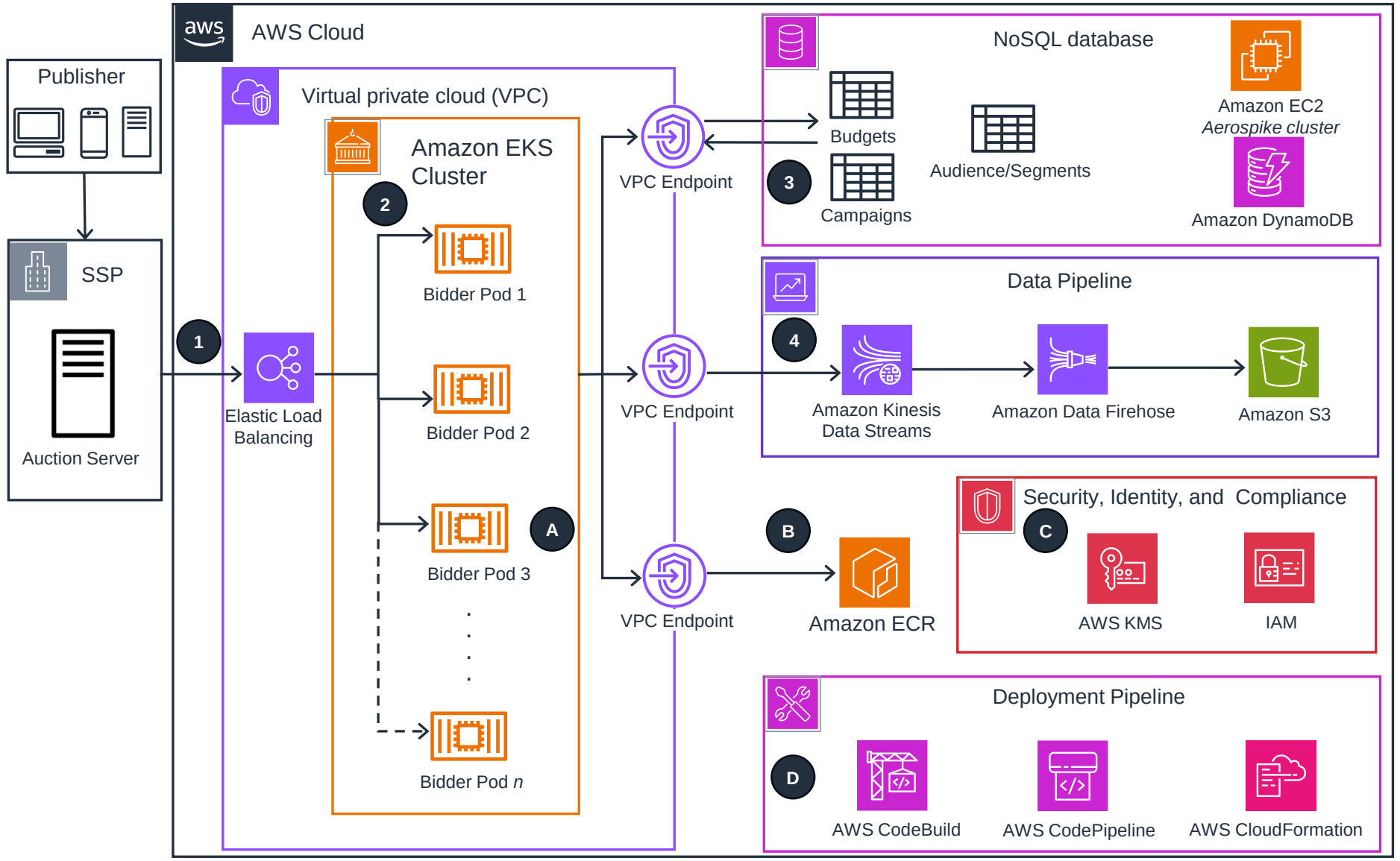
Guidance for Building a Real-Time Bidder for Advertising on AWS

This architecture diagram shows how demand-side partners can efficiently deploy and build upon a stateless open-source architecture. This architecture is optimized for both performance and cost, enabling the rapid assessment of ad opportunities at scale.



1. The supply-side platform (SSP) receives an ad request from a publisher, creates a real-time auction, and sends a bid request to a demand-side public endpoint that is configured on **Elastic Load Balancing**.

2. The requests are routed to "bidder" pods hosted on an **Amazon Elastic Kubernetes Service** (Amazon EKS) cluster. The bidder uses **Amazon Virtual Private Cloud** (Amazon VPC) endpoints to access NoSQL database tables that hold information about audiences/segments, campaigns, and budgets. The bidder uses this information to process the bids.

3. Based on the data from the NoSQL database (such as Aerospike or **Amazon DynamoDB**), the bidder decides whether to bid or not. If a sent bid wins, the bidder updates the budgets and campaign database tables. **NOTE**: Aerospike will run within the **Amazon VPC** and does not require an **Amazon VPC** endpoint. Configure the rack-aware feature on Aerospike for better performance.

4. The bidder transactions are sent to **Amazon Kinesis Data Streams** through **Kinesis Data Streams Amazon VPC** endpoints in compressed micro batches of 25 KB PUTs. **Amazon Data Firehose** then sends this data to **Amazon Simple Storage Service** (Amazon S3) for downstream analytics and reporting. A data stream enables the bidder to respond faster and helps in scaling each component independently.

A. Use AWS Graviton Processor for bidder nodes. For additional cost optimization, implement auto-scaling and **Amazon Elastic Compute Cloud** (Amazon EC2) Spot Instances.

B. Pre-install bidder container images with dependent libraries and binaries to minimize boot time. Upload the images to a container registry, like **Amazon Elastic Container Registry** (Amazon ECR).

C. Encrypt and decrypt data at rest and in transit across **DynamoDB**, **Kinesis Data Streams**, **Amazon EKS**, and **Amazon S3** using **AWS Key Management Service** (AWS KMS). Grant least privilege access using **AWS Identity and Access Management** (IAM) to provide permissions for users, roles, and services.

D. Automate the deployment of the RTB platform using supported git repository, **AWS CodeBuild**, **AWS CodePipeline** and **AWS CloudFormation** to reduce time-consuming, manual processes.

Reviewed for technical accuracy October 30, 2024

AWS Reference Architecture