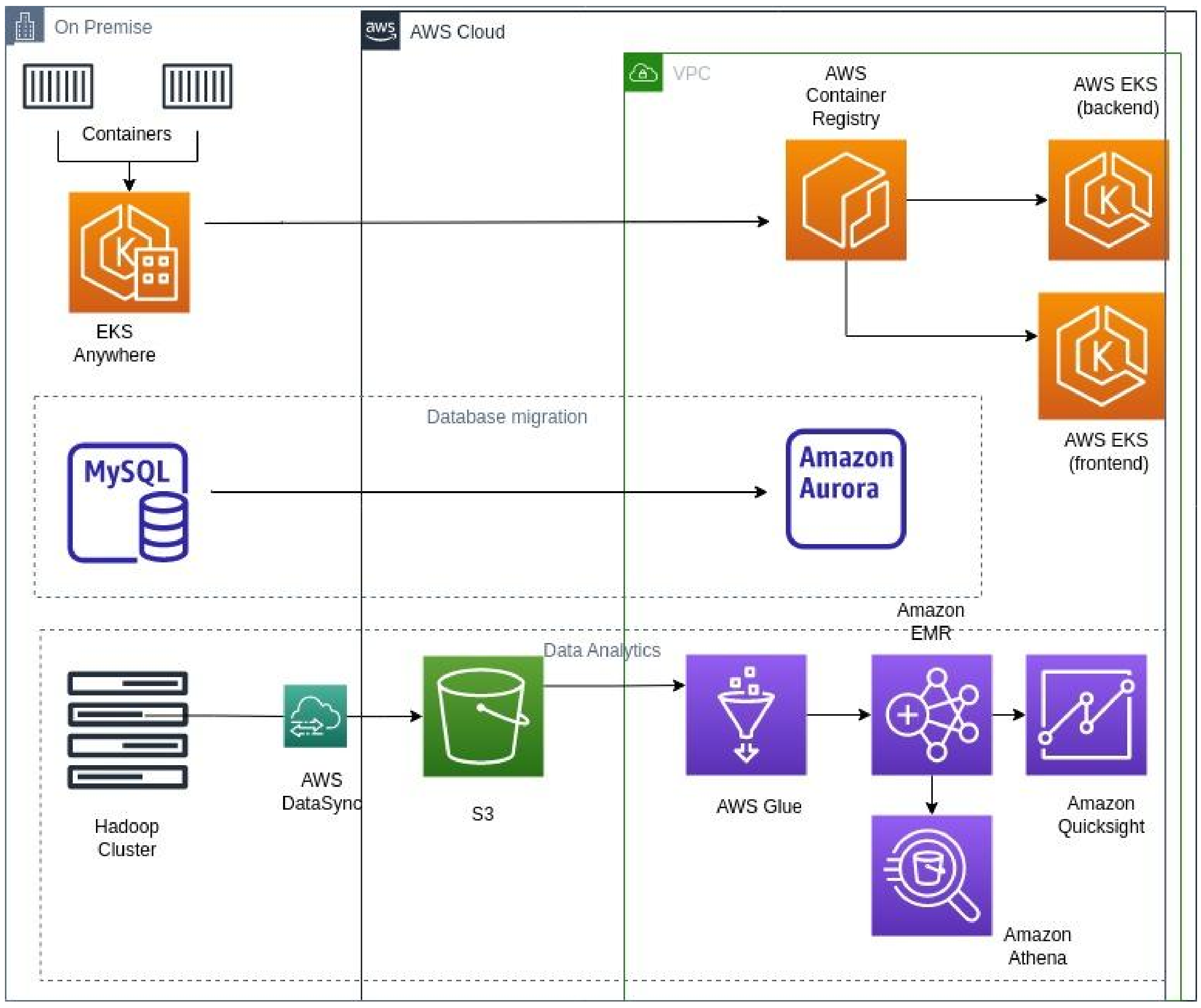
The proposed architecture for the solution is shown in the diagram. In order to make the solution platform agnostic, containerization techniques were suggested as cloud-native techniques to the customer. It also helps to decouple the workload. Despite having an initial technical debt for refactoring the code, in long rung it helps to manage their frontend and backend. Following workflow was used for each of the workload:



1. Frontend: The frontend code (HTML, CSS, JavaScript), was containerized (can use tools like docker) on premise and for the container orchestration, Amazon EKS Anywhere on the on premise side and **Amazon Elastic Kubernetes Service** (Amazon EKS) on AWS Cloud was used. For the registry of the container image, **AWS Container Registry** was used.
2. Backend: The backend code (Java application) and server (Apache Web server) were containerized and Amazon EKS Anywhere on the on premise side and Amazon EKS on AWS Cloud was used.. For the registry of the container image, **AWS Container Registry** was used.
3. Database: MySQL database was migrated using **Amazon Aurora**, which is the serverless managed service from AWS.
4. Hadoop Cluster: For the hadoop cluster, it was first migrated in AWS Cloud using **AWS Data Sync** and stored in **S3.** AWS Glue was used for extract, transform, and

load (ETL) before sending it to **Amazon EMR** for creating clusters. It was possible to query from Amazon EMR using **Amazon Athena**. And finally **Amazon Quicksight** was used for visualization.