**RAK Bank Assessment Documentation**

**EKS :**

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed Kubernetes service to run Kubernetes in the AWS cloud and on-premises data centers. In the cloud, Amazon EKS automatically manages the availability and scalability of the Kubernetes control plane nodes responsible for scheduling containers, managing application availability, storing cluster data, and other key tasks. With Amazon EKS, you can take advantage of all the performance, scale, reliability, and availability of AWS infrastructure, as well as integrations with AWS networking and security services. On-premises, EKS provides a consistent, fully-supported Kubernetes solution with integrated tooling and simple deployment to AWS Outposts, virtual machines, or bare metal servers.

**System Architecture Overview:**

The system architecture comprises a Java Spring Boot application hosted on an Amazon Elastic Kubernetes Service (EKS) cluster. The application's build and deployment processes are orchestrated by a GitHub Actions workflow, while the provisioning of the EKS cluster and associated infrastructure is managed through Terraform. The key components include:

1. **Java Spring Boot Application:**
   * A Java Spring Boot application featuring an API endpoint **/status** for health checks, configured to run on port 8080.
   * The application utilizes Tomcat 8.5 as its embedded servlet container.
2. **GitHub Actions Workflow:**
   * A comprehensive GitHub Actions workflow is responsible for orchestrating the entire Continuous Integration/Continuous Deployment (CI/CD) pipeline.
   * This workflow encompasses all stages, including building the application, running tests, creating Docker images, and deploying to the EKS cluster.
3. **EKS Cluster Deployment:**
   * The EKS cluster is provisioned using modularized Terraform configurations.
   * These Terraform modules are designed to be reusable, facilitating the deployment of multiple EKS clusters with consistent configurations and best practices.

**Challenges:**

* The GitHub Action responsible for publishing artifacts encountered an issue due to different runners being utilized for each job step. Consequently, it was unable to locate the ROOT.War file, which is the Java build artifact. To address this challenge, I initially attempted to consolidate all job steps into a single job. While this approach proved effective, I later optimized the job structure and introduced a dedicated "publish artifact" job. This job is designed to publish the ROOT.war file to the subsequent job, even if it runs on a different runner.
* The Terraform code encountered an issue where it was unable to create the EKS node group due to a missing EC2 trust relationship. To resolve this, I added an EC2 trust relation to the cluster's node group. This adjustment enables the node group to create EC2 instances successfully.
* Public-key generation was performed manually using PuTTYgen, and the generated public key value was added to the authorized keys for any EC2 instance.