**Documentation: Web Application Deployment on EKS with Kubernetes**

**1. Objective**

This project involves deploying a simple web application on Amazon EKS using Kubernetes. The application is containerized using Docker and deployed with Kubernetes Deployment and Service configurations.

**2. Prerequisites**

* AWS CLI installed and configured with proper IAM permissions.
* Docker installed and configured for building container images.
* Terraform for infrastructure management.
* kubectl for managing Kubernetes clusters.
* EKS cluster and Node Group already created.

**3. Steps Followed**

1. **Docker Image Creation**:
   * A Dockerfile was created to containerize the web application.
   * The web application image was pushed to Amazon Elastic Container Registry (ECR).
2. **Kubernetes Configuration**:
   * A Kubernetes Deployment YAML file (deployment.yaml) was created for defining the pod and container configurations.
   * A Kubernetes Service YAML file (service.yaml) was created for exposing the application through a LoadBalancer.
3. **Deployment on EKS**:
   * Kubernetes kubectl was used to apply the configuration files (deployment.yaml and service.yaml) to the EKS cluster.
   * The deployment created two pods for redundancy, and the service exposed the application via a LoadBalancer.
4. **Accessing the Web Application**:
   * The external URL for the LoadBalancer was generated and used to access the deployed web application.

**4. Access URL**

* The static URL of the deployed webpage is:
  + [**http://ab3d6941ae66d45aebed0d4a6ecb5a53-371752692.us-east-1.elb.amazonaws.com**](http://ab3d6941ae66d45aebed0d4a6ecb5a53-371752692.us-east-1.elb.amazonaws.com)

**5. Conclusion**

* The web application was successfully deployed on Amazon EKS.
* Kubernetes managed the deployment and ensured high availability with multiple replicas.
* The service was exposed through an AWS LoadBalancer, making the application accessible via a public URL.