Problem-Statement

Here's the problem statement broken down into points:

Problem Statement: Deploy Java HTTP Server to AKS Using Terraform

1. Write a Java program `HelloWorldHttpServer.java` that creates an HTTP server serving the "Hello, World!" message.
2. Compile the Java program to generate the class files, run this program to test whether code is working locally as expected `http://localhost:8080/`.
3. Create a Dockerfile to build a Docker image for the Java application.
4. Use an appropriate OpenJDK or OpenJRE base image in the Dockerfile to set up the runtime environment.
5. Copy the compiled `HelloWorldHttpServer.class` and the `HelloWorldHttpServer.java` source file into the Docker image.
6. Compile the Java program inside the Docker image (if needed).
7. Expose the appropriate port (e.g., port 8080) in the Dockerfile.
8. Build the Docker image using the `docker build` command.
9. Test the Docker image locally by running a container and accessing the "Hello, World!" message at `http://localhost:8080/`.
10. Choose a container registry (e.g., Docker Hub, Azure Container Registry) to store the Docker image.
11. Authenticate with the container registry (if required).
12. Tag the Docker image with the registry's URL.
13. Push the Docker image to the container registry using the `docker push` command.
14. Write Terraform configuration files to define the AKS cluster.
15. Specify the desired node pool size, Kubernetes version, and other configurations in the Terraform files.
16. Initialize Terraform using `terraform init`.
17. Apply the Terraform configuration to create the AKS cluster using `terraform apply`.
18. Authenticate with the AKS cluster to interact with Kubernetes using the `az` CLI or Kubernetes configuration files.
19. Create a Kubernetes deployment or pod manifest to deploy the Docker image as a Kubernetes Pod or Deployment.
20. Apply the deployment or pod manifest using `kubectl apply` to deploy the application to the AKS cluster.
21. Create a Kubernetes Service manifest to expose the deployed application to the internet using a LoadBalancer or NodePort type.
22. Apply the service manifest using `kubectl apply` to expose the application.
23. Access the public IP or URL of the exposed service to test the "Hello, World!" message in the AKS cluster.

By following these steps, you can successfully deploy the Java HTTP server to an Azure Kubernetes Service (AKS) cluster using Terraform. The application will be accessible on the internet, serving the "Hello, World!" message.

Revision #1
Created 27 July 2023 09:54:56 by Gaurav Pandey
Updated 27 July 2023 09:59:12 by Gaurav Pandey