Readme File

Group Name: Alpha1

Name: Venkata Sathya Gopala Krishna Chada

Prerequisites:

Ensure that Git, Docker, and Kubectl are installed on your local machine. Make sure you have your login credentials saved for GitHub and Docker Hub.

Objective:

Containerize the web application using Docker container technology. Steps to Create and Upload Docker Image:

1. Register at hub.docker.com, log in, and download Docker Desktop.
2. Verify Docker installation by running: `docker -v`
3. Create a project directory named 'docker-project'.
4. In this directory, create a text file named 'Dockerfile' without any extension.
5. Use the 'index.html' file from Homework 1 Part 2 as the survey HTML document, and build the Dockerfile with the following command:
   - `docker build -t hw2-docker-image .`
6. Verify that the image has been created using:
   - `docker images`
7. Tag and push the image to Docker Hub:
   - `docker tag hw2-docker-image rprasad6/hw2-docker-image:1.0`
   - `docker push rprasad6/hw2-docker-image:1.0`
8. Confirm the image upload on Docker Hub.

Dockerfile Example:

```
FROM --platform=linux/amd64 ubuntu:latest
RUN apt-get update
RUN apt-get install -y nginx
RUN rm /var/www/html/index.nginx-debian.html
COPY index.html /var/www/html/
COPY j.jpg /var/www/html/
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Objective:

Deploy the containerized application on Kubernetes to achieve scalability and resiliency. You can use Rancher or managed Kubernetes services like AWS EKS or Google GKE.

Steps to Set Up Rancher/Kubernetes Cluster in AWS Academy Learner Lab:

9. Log into the AWS Learner Lab and go to the EC2 Instances page.
10. Create three EC2 instances with the following settings:
    - Use the Ubuntu Server 22.04 AMI LTS (HVM), SSD Volume Type.
    - Security group allowing inbound traffic on ports 8080, 80, 443, and 22 from anywhere.
    - Assign 30GB storage to each instance.
    - Configure outbound rules to allow all traffic.
11. Assign Elastic IPs to each instance.
12. Connect to instances 1 and 2 to install Docker.
13. Install Docker on both instances using the following commands:
    - `sudo su -`
    - `sudo apt-get update`
    - `sudo apt install docker.io`
14. Start the Rancher server on instance 1.
    - Use the below command to start the Rancherserver
      $ sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
    - Once that install is complete, run "sudo docker ps" to view the current docker instance information and note down the container-ID.
    - In the UI, copy the password command and paste in instance1 console to get the default password to enter in the RancherUI for the first time login. The noted container-ID must be replaced in the below command
      sudo docker logs container-ID 2>&1 | grep "Bootstrap Password:"
    - Use the generated password to login and once login, select the option "Set a specific password to use" to set a custom password for future.
15. Set up Kubernetes on instance 2.
    - Once custom password is set navigate to home and create a new cluster by clicking the create button and selecting custom under "Use existing nodes and create and cluster using RKE". Give the cluster a name and click Create. In the next page click the three checkboxes named etcd, Control Plane and Worker and apply the checkbox – Insecure: Select this to skip TLS verification if your server has a self-signed certificate. This would give a registration command to be pasted on instance2 connect console.
      Command:

curl --insecure -fL https://52.205.139.15/system-agent-install.sh | sudo sh -s - --server https://52.205.139.15 --label 'cattle.io/os=linux' --token 8bv7qtfg2d2rpssjmxrf7s6hrfjsvzv644m7xw9968lnbz4v8tf2b4 --ca-checksum 6565ed5862ecaf0f6946b9073ebed72944c9a23edf2b24f56fe9915c2ade22ae --etcd --controlplane –worker

- Once the command is completed on instance2 connect console, wait in the RancherUI till the cluster state is changed to 'active' from 'updating'. Once the state is active the cluster is ready to deploy

16. Download the kubeconfig file for future use.
17. Create a deployment object using the Docker container image.
    - After that, Click the cluster name -> from left side menu open 'Workloads' and select 'Deployments'.
    - Click on 'Create' button and fill the following fields
      a) Fill the name field for the deployment object.
      b) Increase count in replica field from 1 to 3.
      c) Fill the container field.
      d) Fill Container image field. (it would be the one created in docker)
      e) Click Add Port or Service and give Nodeport in the Service type and 80 in the private container port field. 80 because in the docker file I have used value 80 for EXPOSE. Click Create button.

Access Application:

You can access the application using the following format:

- `<Public DNS IP of instance 2>:<assigned NodePort>`

http://ec2-18-204-234-137.compute-1.amazonaws.com:32702/

Objective:

Set up a CI/CD pipeline with GitHub as the source code repository and Jenkins for automated build and deployment of the application on Kubernetes.

Steps to Configure Jenkins on Instance 3:

18. SSH into instance 3.
19. Install Docker, JDK, and Jenkins:
    - Update the system packages: `sudo apt-get update`
    - Install Docker: `sudo apt-get install -y docker.io`
    - Install Java Development Kit (JDK): `sudo apt install openjdk-11-jre-headless`
    - Add Jenkins repository key: `wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -`
    - Add Jenkins repository: `sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'`
    - Update and install Jenkins: `sudo apt-get update && sudo apt-get install -y jenkins`

20. Start Jenkins service:
    - `sudo systemctl start jenkins`
    - Verify Jenkins is running: `sudo systemctl status jenkins`
21. Access the Jenkins UI at: `<instance 3 public IPv4 address>:8080`
22. Retrieve the initial Jenkins admin password:
    - `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
    - Use the password to log in to Jenkins for the first time.
23. Install the required Jenkins plugins:
    - Navigate to **Manage Jenkins** -> **Manage Plugins**.
    - Install plugins such as Git, Docker Pipeline, and Kubernetes CLI to ensure integration capabilities.
24. Add GitHub and Docker credentials to Jenkins:
    - Go to **Manage Jenkins** -> **Manage Credentials**.
    - Create credentials for GitHub and Docker Hub (use username and password).
25. Configure Jenkins to use Docker:
    - Add Jenkins user to the Docker group to allow Jenkins to run Docker commands: `sudo usermod -aG docker jenkins`
    - Restart Jenkins to apply changes: `sudo systemctl restart jenkins`
26. Create a Jenkins pipeline for CI/CD:
    - From the Jenkins dashboard, click **New Item**.
    - Provide a name for the pipeline (e.g., `cicdPipeline`) and select **Pipeline**.
    - Under **Build Triggers**, select **Poll SCM** and set the schedule to `* * * * *` for every minute.
    - Under **Pipeline**, select **Pipeline script from SCM** and choose **Git** as the SCM.
    - Provide the GitHub repository URL and select the saved credentials.
    - Specify the branch to use (`main`) and the script path (`Jenkinsfile`).
27. Test the Jenkins pipeline:
    - Make a commit to the GitHub repository to trigger the build.
    - Verify that Jenkins automatically builds and deploys the updated application to Kubernetes.

AWS URL of Your Homepage and Deployed Application on Kubernetes:

http://18.204.234.137:32702/