**Program: Python Hit Count**

**Problem:-**

Create a simple web-page hit counter web application using python or Golang.

* Whenever we open/hit the URL, it should return a “Wonderful..
* We have hit <counter> times” and print on the web-page.
* You can make use of Redis or any Database for storing counter value.
* Containerize all the web application related code and build the images using docker and run it on K8s.

**Guidelines:-**

In order to keep this exercise repeatable we have chosen minikube cluster and docker for building images. Selection of OS, configure management and software packages are up to you.

Required Tools:

* Minikube
* Docker

**Solution :-**

**Step 1 (On Local Machine) :**

1. Create python application for Hit count **“app.py”** & **“requirement.txt”** Files.

2. Use **Redis DB** for storing count number.

3. Create docker file for building image use bellow command

**“Docker build -t hit-counter”** - This Command create Docker image in Docker Desktop

4. Create Docker-Compose File for configure docker image and containerize image in docker

**“Docker-Compose -f docker-compose.yml up”** - This Command build and containerize image on Docker Desktop

1. Push image to Docker hub
2. Give tag to the image using bellow Command **“Docker tag hit-counter-myapp kumaryeoley/hit-counter-myapp”**
3. log in into Dockerhub

**“Docker login -u "username" -p "password" docker.io”**

1. Pushing image to Dockerhub

**“Docker push kumaryeoley/hit-counter-myapp”**

6. Install Kubectl CLI and Minikube on system & on docker or VM (Prefer Docker) respectively

7. Create **“deployment.yml” , “service.yml” & “create-pod.yml”** file

8. Run This Command **“kubectl apply -f deployment.yml”** for deployment of application

9. Run This Command **“minikube dashboard”** for visualize minikube deployment dashboard

10. Run application on **localhost** port number

**Step 2 (On AWS Cloud using Terraform & Shell Script) :**

1. Install **Terraform** in system and Initialize
2. Create **“ec2.tf”** file for the infrastructure we need on cloud like instance , security group , VPN, etc.
3. Create **“build\_docker\_image.sh”** file for the command that need to execute on instance.
4. After File create and ready for execute.
5. Run **“terraform init”** this command initialize terraform in back-end with required plugin , etc
6. Run **“terraform plan”** this command shows the plan or instruction which we write in **“ec2.tf”** file.
7. Run **“terraform apply”** this command shows the plan we need to apply on cloud , it ask us confirmation that : **yes** / no (**yes** for apply the code and no for deny ).
8. After above command execute , it will create all the resource and rules we need to create on cloud. We can also check on **AWS cloud** . after execute command it shows that all resource created.
9. After the resource create we can check on ip with port number or we can use DNS also.
10. Run **“terraform destroy”** this command destroy all the resource we create on AWS cloud

**Step 3 (On AWS Cloud using Ansible)** :

1. Create two instance (machines) on AWS , one is master and one is host
2. Install and configure Ansible on Master instance with **“ansible.cfg”** file
3. Install **python3** on Ansible Host instance
4. Write **“ansible.yml”** file (Ansible Playbook) for deploy web application using docker
5. Check Ansible version on instence using **“ansible --version”** Command
6. Use **“ansible Managed\_node1 --list-hosts”** it will show host IP for your inventory file.
7. Use **“ansible Managed\_Node1 -m ping”**  it will show Success with host IP and send reply **pong.**
8. Use **“ansible-playbook ansible.yml”** for run Ansible Playbook.

It shows : PLAY [Managed Node]

Task [Gathering Facts]

Task [Configuring Docker repository ]

Task [Checking Configuration Status ]

Task [Installing Docker]

Task [Checking Install Status]

Task [Start Docker Demon]

Task [Check Pull Status ]

Task [Create a Volume Dir]

Task [Copy Python Code in the Directory]

Task [launching an HTTPD Container]

1. Use **“Systemctl status docker”** to check whether Docker is running or not
2. Use **“ifconfig”** to see IP address of system
3. Open Browser and search for same **IP address with port number**