Assignment-2

Deployment Document

Assignment Scope and definitions:

A development team wants you to deploy this simple web app called Sinatra App, running on Nginx webserver configured with Unicorn load balancer in the environment.

* Come up with setting up Unicorn with Nginx
* come up with a way to deploy this *sinatra app* and write configuration-as-code to deploy it.
* ensure that the instance is locked down and secure
* suggest ways to do ongoing deployments on this application

Prerequisites and Assumptions:

* We need to setup Nginx webserver configured with Unicorn as load balancer on which Sinatra app will be running.
* We are using docker and Kubernetes as preferred orchestration tools here.
* Kubernetes cannot create containers so docker will be installed first and then Kubernetes can be installed to setup the deployment environment for Sinatra app. Please note Kubernetes is also dependent on Docker as container tool.
* Unicorn load balancer is part of Kubernetes deployment only and configured by kubernetes external services and service type: loadbalancer.
* The configuration-as-code recipes are coming from YAMLs file, which are very simple json format and going to handle complete deployment of Sinatra App running on Nginx with Unicorn load balancer.
* The dockerfile is another recipe file that will take care of all the configurations-as-code.
* The docker software is used to build the image that we require to setup Nginx with Unicorn and install Ruby software as well. All these are taken care by a single dockerfile.
* The output of the dockerfile is a image and the name of this image is sinatra-app. This image will be deployed by Kubernetes as a YAML deployment file, which is in fact our recipe file and Sinatra-app is deployed by this YAML file which is a single file that takes care of everything – Ruby and Nginx webserver with Unicorn as load balancer.
* Once deployment is done by executing the YAML file, Sinatra-app is up and running and can be accessed by *“IP-Address:Port”* **e.g** https://172.10.20.30:80

Environment setup:

* AWS EC2 Instance has to be configured with AMI as Ubuntu 16.04 or 18.04 LTS version.
* EC2 Type: t2.medium :: 2 vCPU and 4 GB of RAM. Spin up 3 EC2 Instances
* Docker Version: 18.0 or 19.0. either will work.
* Kubernetes version: 1.16 will work. Kubernetes Installation steps are provided at the end.
* Create locked down and secure Instance:
  + By configure in-bound rule in security group and add in-bound port number 80 and do not put all-all so it must not be 0.0.0.0/0
  + Configure security groups to permit the minimum required network traffic for the EC2 instance.
  + Create a VPC and IAM role and define the policy to restrict IAM user.
  + This policy restricts an IAM user or group access to only Start/Stop/Reboot EC2 instances in that particular region.
  + Note: Replace the Owner, Bob, and [AWS Region](https://docs.aws.amazon.com/general/latest/gr/rande.html#endpoint-tables) with parameters from your environment.
  + Finally, create similar policies for each group of IAM users, using a different Region for each one.

Solution Approach:

* We have taken Docker and Kubernetes as the preferred orchestration softwares in regards to complete this assignment.
* We have written *Dockerfile* for capturing all the configurations and dependent steps for covering up dependencies that are required for Sinatra-app to be deployed on Nginx web server. So *Dockerfile* is going to be our configuration-as-code recipe for building Sinatra application.
* We have written deployment scripts and here also our configuration-as-code recipes are nothing but *YAML* file. By using a single *YAML* files that will do the complete deployment of Sinatra app running on Nginx web server and Unicorn as load balancer in the Kubernetes cluster.

Configuration-as-code recipes: (*YAML* file)

* Deployment recipe for Nginx with (Unicorn)
* hLoadBalancer Service
* This *YAML* file deploys Sinatra App in the cluster
* Sinatra-App is up and running and can be access by browser.

apiVersion: apps/v1

kind: Deployment

metadata:

name: sinatra-app

labels:

app: sinatra-app

type: front-end

spec:

template:

metadata:

name: sinatra-pod

labels:

app: sinatra-pod

typs: front-end

spec:

containers:

- name: mycontainer

image: my-sinatra-app:1.0

replicas: 3

selector:

matchLabels:

type: front-end

...

apiVersion: v1

kind: Service

metadata:

name: unicorn-lb

labels:

app: sinatra-app

type: front-end

spec:

selector:

app: sinatra-app

type: front-end

type: LoadBalancer

ports:

- port: 80

targetPort: 80

Steps to complete:

* Login to all three AWS EC2 instances by using putty or mobaxterm client
* After login change the default user to root user for docker installation, use command as:

**“sudo su -“**

* Assuming docker and Kubernetes already installed here by following the steps mentioned in the Annexure.
* On Command prompt verify that docker and Kubernetes are properly installed by commands: **“kubectl version –short”**
* Type the command in order to complete the deployment by executing below command:

**“kubectl create -f sinatra-app.definition.yaml”**

* Verify that Pod has been created and Sinatra-app is up and running inside these pods:

**“kubectl get pods” OR “kubectl get all”**

* To access Sinatra Application goto browser and type the public IP-Address of EC2 Instance along with the port:80

Browser URL: ***“https://50.80.90.100:80”***

***Annexure***

*Kubernetes cluster creation: Installation Procedure*

# Install Docker CE

## Set up the repository:

### Install packages to allow apt to use a repository over HTTPS

apt-get update && apt-get install -y apt-transport-https ca-certificates curl software-properties-common

### Add Docker’s official GPG key

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -

### Add Docker apt repository.

add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) \

stable"

## Install Docker CE.

apt-get update && apt-get install -y docker-ce=18.06.2~ce~3-0~ubuntu

# Setup daemon.

cat > /etc/docker/daemon.json <<EOF

{

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

EOF

mkdir -p /etc/systemd/system/docker.service.d

# Restart docker.

systemctl daemon-reload

systemctl restart docker

$docker --version

========================= KUBERNETES INSTALLATION PROCESS ========================

## Now Install Kubeadm

apt-get update && apt-get install -y apt-transport-https curl

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF

apt-get update && apt-get install -y kubelet kubeadm kubectl

systemctl daemon-reload

systemctl restart kubelet

#Install POD Network

kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=$(hostname)

POD Network Type: FLANNEL

kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/62e44c867a2846fefb68bd5f178daf4da3095ccb/Documentation/kube-flannel.yml>

1.2 For more details on Unicorn Loadbalancer:

Reference URL: <https://sirupsen.com/setting-up-unicorn-with-nginx/>