**PG DO - DevOps Capstone Project**

**Olufemi Amosu**

**nicefemi@yahoo.com**

[*https://github.com/oamosu14/capstone\_project\_nicefemi*](https://github.com/oamosu14/capstone_project_nicefemi)

**Implementation requirements:**

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)
2. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet
3. Install Docker and Kubernetes on the cluster
4. Implement the network policies at the database pod to allow ingress traffic from the front-end application pod
5. Create a new user with permissions to create, list, get, update, and delete pods
6. Configure application on the pod
7. Take snapshot of ETCD database
8. Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured

**Below are the task executed:**

* Setup a Local CentOS 7 server as Ansible Controller Node
* Install and configure Ansible on this local server and perform the following tasks:
* Create AWS user
* Install Ansible and Ansible EC2 module dependencies
* Create SSH keys
* Create Ansible structure
* Run Ansible to provision the EC2 instance
* Connect to the EC2 instance via SSH

**Create AWS IAM User**

* Add a user to IAM and give it programmatic access
* Set permissions by adding user to group:
* Take note of the Access Key ID and Secret Access Key that will be used by Ansible to setup the EC2 Instances.
* Create AWS Key Pair and save it as a pem file.
* Download file and change file permission

**Setup Ansible for Kubernetes on AWS**

* Create an ansible.cfg file and enter the following configuration:
* Creating Ansible Vault to store the AWS Credential
* Set the password for the file and enter the Access Key ID and Secret Access Key generated in AWS.
* Create a roles folder and change directory to the path.
* Edit the main.yml file generated by ansible-galaxy within each roles folder accordingly.
  + Code for K8s\_master
  + Code for k8s\_worker
* Add a setup.yml file in the project directory and add content as follows:
* Run “ansible-playbook setup.yml --ask-vault-pass” to create a 3-node Kubernetes cluster on AWS.
* Enter the password created for the vault.

**This command implements the 3 first requirements in the Capstone Project:**

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)
2. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet
3. Install Docker and Kubernetes on the cluster

**Implement Network Policy**

* Add a yaml file and append:
* Enter the command “kubectl apply -f network\_policy.yaml”

**Create new User with permissions to create, list, get, update, and delete pods**

* Create a private key
* openssl genrsa -out myuser.key 2048
* Generate a CertificateSigninRequest CSR
* openssl req -new -key myuser.key -out myuser.csr -subj "/CN=myuser"
* The certificate request we'll use in the CertificateSigningRequest
* The CertificateSigningRequest needs to be base64 encoded
* Submit the CertificateSigningRequest to the API Server
* Approve the CSR
* Retrieve and save the certificate in the csr object in Kubernetes to file:
* Create a kubeconfig file for the user myuser using the generated crt and key files:
* Set the cluster
  + Set the credentials
  + Set context:

kubectl config set-context myuser@kubernetes-demo \

--cluster=kubernetes \

--user=myuser \

--kubeconfig=myuser.conf

* + Create a ClusterRoleBinding Object to bind the myuser to a Role/ClusterRole. (I used the default view for initial testing)
  + create a new linux user and then create a new kubeconfig for that user:
  + Copy the myuser.conf kubeconfig to the home directory of demo user in the default kubeconfig location of .kube/config
  + Create a clusterRole to allow create,list,get,update,delete permissions on pod resource:
* Create a ClusterRoleBinding to attach ClusteRole to myuser:

**Configure Application pods**

* Creating the Redis (Backend) Deployment:
* Creating the Redis (Backend) Service:
* Create the frontend Deployment:
* Create the frontend service:
* Apply configuration using kubectl apply -f file\_name.yaml

**Take Snapshots**

* Get etcd pod
* Get release version of etcd
* Download etcdctl binary an extract to bin folder:
* Define a variable for the endpoint to etcd:
* ENDPOINT=https://127.0.0.1:2379
* Take the backup saving it to /var/lib/dat-backup.db

**Configure Horizontal Pod Autoscaler**

* Deploy a metric server. But first download the yaml file:
* wget https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
* Edit the yaml file by adding command block in the pod spec:
* Apply configuration to deploy metric server.
* Create yaml config file for Horizontal Pod Autoscaler:
* Deploy using kubectl apply -f frontend-ruby-hpa.yaml
* Simulate high workload to test it

**SNAPSHOTS**

**Setting up AWS**

* Add a user to **IAM** and give it **programmatic access**

Graphical user interface, text, application, email

Description automatically generated

* Set permissions by adding user to group:

Text

Description automatically generated with low confidence

Graphical user interface, text, email

Description automatically generated

* Make sure you save the **Access Key ID** and **Secret Access Key**

Create AWS **Key Pair**  and save it as a pem file.

Graphical user interface, application

Description automatically generated

**Setup Ansible for Kubernetes on AWS**

* Create an **ansible.cfg** file and enter the following configuration:

A picture containing text, monitor, screenshot

Description automatically generated

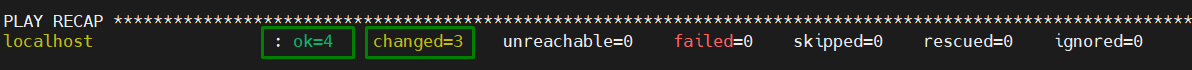
Text

Description automatically generated

* Ansible Playbook to provision EC2 Intances

Text

Description automatically generated



* EC2 Dashboard

Graphical user interface, text, application, website

Description automatically generated

* Ansible Playbook to provision EC2 Instances

Text

Description automatically generated

* Docker Installation

A picture containing text, outdoor

Description automatically generated

**Setting up Kubernetes Cluster**

* Using Ansible playbook

A picture containing table

Description automatically generated

**Implement Network Policy**

To implement a network policy

* Add a yaml file and append:
* Enter the command **“kubectl apply kube-files/network\_policy.yaml”**

Text

Description automatically generated

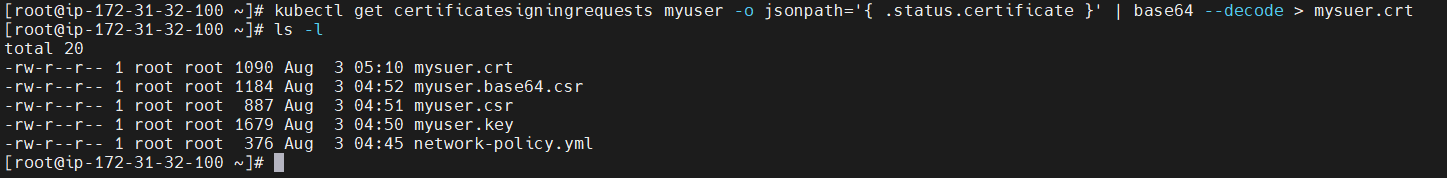
Text

Description automatically generated

**Create new User with permissions to create, list, get, update, and delete pods**

Text

Description automatically generated

* Approve the CSR
* Retrieve and save the certificate in the csr object in Kubernetes to file:
* 
* Create a **kubeconfig** file for the user **myuser** using the generated **crt** and **key** files**:**
* Set the cluster

Text

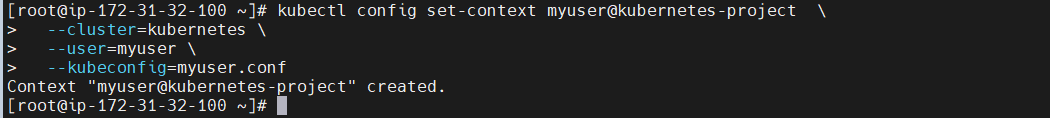
Description automatically generated

* Set the credentials

Text

Description automatically generated

* Set the context:



kubectl config set-context myuser@kubernetes-demo \

--cluster=kubernetes \

--user=myuser \

--kubeconfig=myuser.conf

* Create a **ClusterRoleBinding** Object to bind the myuser to a Role/ClusterRole. (I used the default view for initial testing)
* 
* create a new linux user and then create a new kubeconfig for that user:
* Text

  Description automatically generated
* Copy the myuser.conf kubeconfig to the home directory of demo user in the default kubeconfig location of .kube/config

Text

Description automatically generated

* Create a clusterRole to allow create,list,get,update,delete permissions on pod resource:
* Create a ClusterRoleBinding to attach ClusteRole to myuser:

Text

Description automatically generated

* Apply yaml file:

**Configure Application pods**

* Creating the Redis (Backend) Deployment:

A picture containing text

Description automatically generated

Text

Description automatically generated

* Creating the Redis (Backend) Service:

Text

Description automatically generated

* Create the frontend Deployment:

Text

Description automatically generated

* Create the frontend service:

Text

Description automatically generated

* Apply configuration using **kubectl apply -f *file\_name.yaml***

**Take Snapshots**

* Get etcd pod

A picture containing graphical user interface

Description automatically generated

* Get release version of etcd

kubectl exec -it etcd-kubemaster -n kube-system -- /bin/sh -c 'ETCDCTL\_API=3 /usr/local/bin/etcd --version' heard



* Download etcdctl binary an extract to bin folder:

A picture containing timeline

Description automatically generated

export RELEASE="3.4.13"

wget https://github.com/etcd-io/etcd/releases/download/v${RELEASE}/etcd-v${RELEASE}-linux-amd64.tar.gz

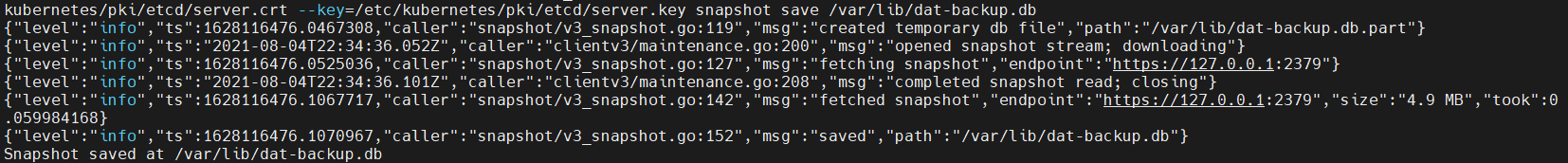
tar -zxvf etcd-v${RELEASE}-linux-amd64.tar.gz

cd etcd-v${RELEASE}-linux-amd64

sudo cp etcdctl /usr/local/bin

* Define a variable for the endpoint to etcd:

ENDPOINT=https://127.0.0.1:2379



* Take the backup saving it to /var/lib/dat-backup.db

Text

Description automatically generated

**Configure Horizontal Pod Autoscaler**

* Deploy a metric server. But first download the yaml file:
* wget <https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml>
* Edit the yaml file by adding command block in the pod spec:

A picture containing graphical user interface

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

Text

Description automatically generated

* Apply configuration to deploy metric server.
* Create yaml config file for Horizontal Pod Autoscaler:

Text

Description automatically generated

* Deploy using **kubectl apply -f frontend-ruby-hpa.yaml**

Graphical user interface

Description automatically generated with low confidence

Graphical user interface

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

==========================================================================================================================================================================================================================================

* Simulate high workload by running

while true; do wget -q -O- http://nginx-service.default.svc.cluster.local; done

Text

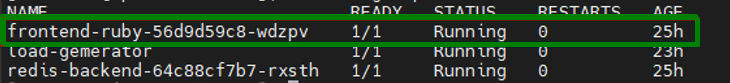
Description automatically generated

* After some seconds, number of pods is automatically scaled up:

A picture containing text, green, grass, outdoor

Description automatically generated

* Stop the job and after some time the pods are automatically scaled down:



Below is my github repository for the source codes:

https://github.com/oamosu14/capstone\_project\_nicefemi