Github URL

https://github.com/estephanj99/Assinment-3/tree/Assignment3

The commands we used for creating secrets for our assignment is *$ kubectl create secret generic admin-login-secrets --from-literal=username=admin --from-literal=password=thisisatestthing. --from-literal=db\_pass=thisisatestthing.*

For fetching the outputs to put in Django-secrets.yaml *$ kubectl get secret admin-login-secrets -o yaml.*

*As mentioned in the assignment instruction’s part 1, “There are many values that are supposed to be secret that are floating around in the source code* *and the yaml files”, which we see in this assignment from the Shoddycorp, “normally we don’t want to see this. Secret values should be protected so that we can move the source code GitHub and place the docker images on dockerhub and not compromise any secrets…” So we are using the secret files to secure our secrets in yaml file in the deployment. We can use this file and we can directly fetch secrets from admin-login-secrets:*

- name: MYSQL\_ROOT\_PASSWORD

valueFrom:

secretKeyRef:

name: admin-login-secrets

key: db\_pass

- name: ADMIN\_UNAME

valueFrom:

secretKeyRef:

name: admin-login-secrets

key: username

- name: ADMIN\_PASS

valueFrom:

secretKeyRef:

name: admin-login-secrets

key: password

For us to seed the database we created a yaml file called db-seed.yml, which contains:

*apiVersion: batch/v1*

*kind: Job*

*metadata:*

*name: database-seeding*

*spec:*

*template:*

*spec:*

*containers:*

*- name: mysql-container*

*image: nyuappsec/assign3-db:v0*

*env:*

*- name: MYSQL\_ROOT\_PASSWORD*

*valueFrom:*

*secretKeyRef:*

*name: admin-login-secrets*

*key: password*

*command: ["/bin/bash"]*

*args:*

*- "-c"*

*- |*

*echo 'Running setup script...'*

*mysql -u root -p${MYSQL\_ROOT\_PASSWORD} -D GiftcardSiteDB -h mysql-service < /docker-entrypoint-initdb.d/setup.sql*

*echo 'Finished running setup script'*

*restartPolicy: Never*

*backoffLimit: 4*

Then we run

$ kubectl apply -f db/k8

We modified the docker file to add

CMD ["mysqld", "--secure-file-priv=/data"]

At End Of File.

To seed the job we used the following code inside the docker container:

$ mysql -u root -p${MYSQL\_ROOT\_PASSWORD} -D GiftcardSiteDB -h mysql-service < /docker-entrypoint-initdb.d/setup.sql

This command will sed the database and put the setup.sql in the required place.

To migrate the job we fetched the container and run the container in the kubeconfig.yml file

apiVersion: batch/v1

kind: Job

metadata:

name: mysql-migration-job

spec:

template:

spec:

containers:

- name: mysql-migration-job

image: nyuappsec/assign3-db-migrations:v0

resources: {}

volumeMounts:

- name: mysql-volume-mount

mountPath: /var/lib/mysql

restartPolicy: OnFailure

volumes:

- name: mysql-volume-mount

persistentVolumeClaim:

claimName: mysql-pvc

status: {}

Here image called nyuappsec/assign3-db-migrations:v0 is fetched.

We decided to remove all monitoring that exposed any sensitive secrets, we also added a Prometheus counter that counts all the times we purposely return a 404 message in the Views.py as the instructions explains. As explained all the changes will be done in the Views.py file:

import prometheus\_client

from prometheus\_client.core import CollectorRegistry

from prometheus\_client import Summary, Counter, Histogram, Gauge

from django.contrib.auth.hashers import make\_password

from django.core.files.base import ContentFile

from django.core.files.storage import default\_storage

To install Prometheus to be able to add the counter we used the commands:

*$ helm repo add prometheus-community* [*https://prometheus-community.github.io/helm-charts*](https://prometheus-community.github.io/helm-charts)

*$ helm repo update*

*$ kubectl create namespace prometheus*

*$ helm install prometheus prometheus-community/kube-prometheus-stack --namespace Prometheus*

We waited for approximately 5 minutes…

And now to configure Prometheus we used the commands:

*$ kubectl port-forward -n prometheus prometheus-prometheus-kube-prometheus-prometheus-0 9090*

*and*

*$ minikube service proxy-service*