Part 1

I started Part 1 by doing reconnaissance for Kubernetes YAML files which have the word ‘SECRET’ in them. Accordingly, I ran “grep -r 'secret' .” in the main directory to recursively find all files with the keyword ‘secret’ in them.

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3$ grep -r 'secret' .
./HW3_Instructions.md:Unfortunately there are many values that are supposed to be secret floating
./HW3_Instructions.md:and put the docker images on Dockerhub and not compromise any secrets. In
./HW3_Instructions.md:addition to keeping secrets secret, this method also allows for changing secrets
./HW3_Instructions.md:For this part, your job will be to find some the places in which secrets are
./HW3_Instructions.md:used and replace them with a more secure way of doing secrets. Specifically, you
./HW3_Instructions.md:should look into Kubernetes secrets, how they work, and how they can be used
./HW3_Instructions.md:1. All kubernetes yaml files modified to use secrets
./HW3_Instructions.md:   settings.py as mentioned above) needed to use the passed secrets.
./HW3_Instructions.md:3. A file, called secrets.txt, which demonstrates how you added the secrets.
./HW3_Instructions.md:remain secret!
./HW3_Instructions.md:monitoring that exposes any sensitive secrets.
./HW3_Instructions.md:* 20 points for the yaml files that use Kubernetes secrets.
./HW3_Instructions.md:  * Your yaml files using Kubernetes secrets.
./HW3_Instructions.md:  * A writeup called secrets.txt.
Binary file ./kubectl matches
./db/k8/db-deployment.yaml:            - name: securedsecrets
./securedsecrets.yaml:    name: securedsecrets
./securedsecrets.yaml:    secret_key: a21neXNhI2Z6KzkoejEqPWMweWRyaml6ayo3c3RobTJnYTF6ND1eNjEkY3hjcThiJGw=
./GiftcardSite/templates/fonts/icomoon/style.css:.icon-user-secret:before {
./GiftcardSite/templates/fonts/icomoon/fonts/icomoon.svg:<glyph unicode="&#xf21b;" glyph-name="user-secret" horiz-adv-x="805" d="M329.143 73.143l54.857 256-5
143 365.714-73.143-36.571-54.857-73.143zM566.857 650.286c-0.571 1.143-1.143 2.286-2.286 3.429-5.143 4-46.286 4.571-54.857 4.571-32.571 0-63.429-4.571-95.429-
-32 6.286-62.857 10.857-95.429 10.857-8.571 0-49.714-0.571-54.857-4.571-1.143-1.143-1.714-2.286-2.286-3.429 0.571-5.143 1.143-10.286 2.286-15.429 3.429-4.571
72 73.143-72 73.714 0 53.143 68 77.143 68h6.857c24 0 3.429-68 77.143-68 51.429 0 58.286 31.429 73.143 72 2.286 6.857 5.143 5.143 8.571 9.714 1.143 5.143 1.71
.143-148-152.571-148h-499.429c-91.429 0-152.571 54.857-152.571 148 0 103.429 18.286 260 124.571 311.429l-51.429 125.714h122.286c-8 23.429-12.571 48-12.571 73
4.571-110.857 22.857-110.857 54.857 0 33.714 97.143 52 120 56.571 12 42.857 40.571 108 69.714 141.714 11.429 13.143 25.714 21.143 43.429 21.143 34.286 0 61.7
.714 0 32-8 43.429-21.143 29.143-33.714 57.714-98.857 69.714-141.714 22.857-4.571 120-22.857 120-56.571 0-32-88.571-50.286-110.857-54.857 2.857-30.857-1.143-
02.286-53.143 120-206.857 120-308.571z" />
./GiftcardSite/templates/fonts/icomoon/selection.json:          "user-secret"
./GiftcardSite/templates/fonts/icomoon/selection.json:        "name": "user-secret",
./GiftcardSite/templates/fonts/icomoon/demo.html:                <span class="icon-user-secret">
./GiftcardSite/templates/fonts/icomoon/demo.html:                <span class="mls"> icon-user-secret</span>
./GiftcardSite/k8/django-deploy.yaml:                secretKeyRef:
./GiftcardSite/k8/django-deploy.yaml:                  name: securedsecrets
./GiftcardSite/k8/django-deploy.yaml:                secretKeyRef:
./GiftcardSite/k8/django-deploy.yaml:                    name: admin-login-secrets
./GiftcardSite/k8/django-deploy.yaml:                secretKeyRef:
./GiftcardSite/k8/django-deploy.yaml:                    name: admin-login-secrets
./GiftcardSite/k8/django-deploy.yaml:                secretKeyRef:
./GiftcardSite/k8/django-deploy.yaml:                  name: securedsecrets
./GiftcardSite/k8/django-deploy.yaml:                  key: secret_key
./GiftcardSite/k8/django-admin-pass-secret.yaml:    name: admin-login-secrets
./GiftcardSite/LegacySite/views.py:        #FIX: commenting this out as tthis may reveal secrets during login process in debug logs
./GiftcardSite/GiftcardSite/settings.py:#SECURITY WARNING: keep the secret key used in production secret!
./GiftcardSite/GiftcardSite/settings.py:SECRET_KEY = os.environ.get('SECRET_KEY') #storing secret as local var in secrets.yaml file
Binary file ./.git/index matches
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3$
```

After closer inspection of the results, I found that the following files contained secrets that were hard-coded or otherwise exposed.

./GiftcardSite/k8/django-deploy.yaml
./GiftcardSite/LegacySite/views.py
./GiftcardSite/GiftcardSite/settings.py
./db/k8/db-deployment.yaml
/db/k8s/db-deployment.yaml

In the views.py file, I commented out a function that would expose the password in the debug logs, as suggested by Kevin's earlier comment.

```
        graphs['r_counter'].inc()
        context = {'method':'POST'}
        uname = request.POST.get('uname', None)
        pword = request.POST.get('pword', None)

        # KG: Uh... I'm not sure this makes sense.
        # Collect data to ensure good password use.
        #FIX: commenting this out as this may reveal secrets during login process in debug logs

        #if pword not in graphs.keys():
        #    graphs[pword] = Counter(f'counter_{pword}', 'The total number of '\
        #      + f'times {pword} was used')
        graphs[pword].inc()
```

In the settings.py file, I commented out the SECRET_KEY and used an environmental variable instead to store the secret locally.

settings.py
~/Documents/appsec/AppSecAssignment3/GiftcardSite/GiftcardSite

```
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

#SECURITY WARNING: keep the secret key used in production secret!
#SECRET_KEY = 'kmgysa#fz+9(z1*=c0ydrjizk*7sthm2ga1z4=^61$cxcq8b$l'
#
SECRET_KEY = os.environ.get('SECRET_KEY') #storing secret as local var in secrets.yaml file

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = bool(int(os.environ.get('DEBUG', 0)))

ALLOWED_HOSTS = ['*']
#ALLOWED_HOSTS_ENV = os.environ.get('ALLOWED_HOSTS')
#if ALLOWED_HOSTS_ENV:
#    ALLOWED_HOSTS.extend(ALLOWED_HOSTS_ENV.split(','))
```

This brought me to django-deploy.yaml and db-deployment.yaml files which both had hardcoded secrets. In order to resolve this, I followed the documentation from this article (https://betterprogramming.pub/how-to-use-kubernetes-secrets-for-storing-sensitive-config-data-f3c5e7d11c15?gi=e1fcd44dcef3) which suggested that I create a separate yaml file to store these secrets and then simply link them back via Reference in each of the yaml files above. Insofar as template for the new yaml file is concerned, I followed the existing 'django-admin-pass-secret.yaml' format and simply renamed it to 'securedsecrets.yam;' and added the missing SECRET_KEY which was commented out from the settings.py file.
For each of the secrets, I ensure they were base64 encoded by running
'echo <key> | base64'

securedsecrets.yaml - Visual Studio Code

File Edit Selection View Go Run Terminal Help

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

! django-deploy.yaml | ! db-deployment.yaml ~/.../k8 | ! db-deployment.yaml ~/.../k8s | ! securedsecrets.yaml × | ! django-admin-pass-secret.yaml

home > ubuntu > Documents > appsec > AppSecAssignment3 > ! securedsecrets.yaml

```
apiVersion: v1
kind: Secret
metadata:
    name: securedsecrets
type: Opaque
data:
    username: YWRtaW4=
    password: dGhpc2lzYXRlc3R0aGluZy4=
    secret_key: a21neXNhI2Z6KzkoejEqPWMweWRyaml6ayo3c3RobTJnYTF6ND1eNjEkY3hjcThiJGw=
```

Next, I applied the updated secured secrets file by running ‘kubectl apply -f securedsecrets.yaml’

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part1$ kubectl apply -f s
ecuredsecrets.yaml
secret/securedsecrets created
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part1$
```

Additionally, I ran ‘kubectl get secrets’ which confirmed that securedsecrets were indeed applied.

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/GiftcardSite/LegacySite$ kubectl get secrets
NAME                  TYPE     DATA   AGE
admin-login-secrets   Opaque   2      16d
securedsecrets        Opaque   3      15h
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/GiftcardSite/LegacySite$
```

Finally, I updated each of the above yaml files to link back to the secrets file, thus removing the hard-coded secret exposure

/db/k8s/db-deployment.yaml file:

```
home > ubuntu > Documents > appsec > AppSecAssignment3 > db > k8s > ! db-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-container
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql-container
  template:
    metadata:
      labels:
        app: mysql-container
        tier: backend
    spec:
      containers:
        - name: mysql-container
          image: nyuappsec/assign3-db:v0
          env:
            - name: securedsecrets
              value: password

            - name: MYSQL_DB
              value: GiftcardSiteDB
```

./db/k8/db-deployment.yaml

home > ubuntu > Documents > appsec > AppSecAssignment3 > db > k8 > ! db-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-container
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql-container
  template:
    metadata:
      labels:
        app: mysql-container
        tier: backend
    spec:
      containers:
        - name: mysql-container
          image: nyuappsec/assign3-db:v0
          env:
            - name: securedsecrets
              value: password

            - name: MYSQL_DATABASE
              value: GiftcardSiteDB

          ports:
            - containerPort: 3306
          volumeMounts:
            - name: mysql-volume-mount
              mountPath: /var/lib/mysql

      volumes:
        - name: mysql-volume-mount
          persistentVolumeClaim:
            claimName: mysql-pvc

```

./GiftcardSite/k8/django-deploy.yaml

File Edit Selection View Go Run Terminal Help

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

! django-deploy.yaml × ! db-deployment.yaml ~/.../k8 ! db-deployment.yaml ~/.../k8s ! securedsecrets.yaml ! django-admin-pass

home > ubuntu > Documents > appsec > AppSecAssignment3 > GiftcardSite > k8 > ! django-deploy.yaml

```yaml
    template:
      metadata:
        labels:
          pod: assignment3-django-deploy
      spec:
        containers:
          - name: assignment3-django-deploy
            image: nyuappsec/assign3:v0
            ports:
              - containerPort: 8000
            env:
              - name: MYSQL_ROOT_PASSWORD
                valueFrom:
                  secretKeyRef:
                    name: securedsecrets
                    key: password

              - name: MYSQL_DB
                value: GiftcardSiteDB

              - name: MYSQL_HOST
                value: mysql-service

              - name: ALLOWED_HOSTS
                value: "*,"

              - name: ADMIN_UNAME
                valueFrom:
                  secretKeyRef:
                      name: admin-login-secrets
                      key: username

              - name: ADMIN_PASS
                valueFrom:
                  secretKeyRef:
                      name: admin-login-secrets
                      key: password

              - name: SECRET_KEY
                valueFrom:
                  secretKeyRef:
                    name: securedsecrets
                    key: secret_key

            volumeMounts:
              - name: mysql-volume-mount
                mountPath: /var/lib/busybox

              - name: static-data-volume-mount
                mountPath: /vol/static

        volumes:
```

Finally, in order to make sure the update propagated accordingly as per documentation from kubernetes (https://www.containiq.com/post/using-kubectl-to-restart-a-kubernetes-pod), I deleted each of the pods so the settings can refresh for each of the containers running
I did this by running ‘kubectl delete pod <insert pod name>’

```
ubuntu@ubuntu2004: ~/Documents/appsec/AppSecAssignment3/GiftcardSite/LegacySite
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/GiftcardSite/LegacySite$ kubectl get pods
NAME                                        READY   STATUS    RESTARTS        AGE
assignment3-django-deploy-5df896c8f-tnbd6   1/1     Running   0               12m
mysql-container-ff8844dc9-sm267             1/1     Running   0               11m
proxy-89c4bc9c5-dx7gq                       1/1     Running   19 (32m ago)    15h
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/GiftcardSite/LegacySite$ kubectl delete pod proxy-89c4bc9c5-dx7gq
pod "proxy-89c4bc9c5-dx7gq" deleted
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/GiftcardSite/LegacySite$ kubectl delete pod mysql-container-ff8844dc9-sm267
pod "mysql-container-ff8844dc9-sm267" deleted
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/GiftcardSite/LegacySite$ kubectl delete pod assignment3-django-deploy-5df896c8f-tnbd6
pod "assignment3-django-deploy-5df896c8f-tnbd6" deleted
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/GiftcardSite/LegacySite$
```

Lastly, I checked the container's env file to ensure the keys were propagated accordingly.

I did this by running ‘kubectl exec -it <container name> /bin/sh’

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/GiftcardSite/LegacySite$ kubectl exec -it mysql-container-ff8844dc9-gs64z /bin/sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
sh-4.4# env
MYSQL_SERVICE_PORT_3306_TCP=tcp://10.102.210.234:3306
PROXY_SERVICE_PORT_8080_TCP_PROTO=tcp
securedsecrets=password
ASSIGNMENT3_DJANGO_SERVICE_PORT_8000_TCP=tcp://10.104.70.181:8000
ASSIGNMENT3_DJANGO_SERVICE_PORT_8000_TCP_PROTO=tcp
ASSIGNMENT3_DJANGO_SERVICE_PORT=tcp://10.104.70.181:8000
PROXY_SERVICE_PORT_8080_TCP_ADDR=10.103.187.108
HOSTNAME=mysql-container-ff8844dc9-gs64z
PROXY_SERVICE_PORT_8080_TCP_PORT=8080
MYSQL_DATABASE=GiftcardSiteDB
PROXY_SERVICE_SERVICE_PORT=8080
KUBERNETES_PORT_443_TCP_PROTO=tcp
MYSQL_SERVICE_PORT=tcp://10.102.210.234:3306
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
PROXY_SERVICE_PORT_8080_TCP=tcp://10.103.187.108:8080
PROXY_SERVICE_PORT=tcp://10.103.187.108:8080
ASSIGNMENT3_DJANGO_SERVICE_SERVICE_PORT=8000
KUBERNETES_PORT=tcp://10.96.0.1:443
MYSQL_SERVICE_SERVICE_HOST=10.102.210.234
ASSIGNMENT3_DJANGO_SERVICE_SERVICE_HOST=10.104.70.181
PWD=/
HOME=/root
MYSQL_MAJOR=8.0
GOSU_VERSION=1.14
KUBERNETES_SERVICE_PORT_HTTPS=443
MYSQL_SERVICE_PORT_3306_TCP_ADDR=10.102.210.234
KUBERNETES_PORT_443_TCP_PORT=443
MYSQL_VERSION=8.0.31-1.el8
ASSIGNMENT3_DJANGO_SERVICE_PORT_8000_TCP_ADDR=10.104.70.181
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
MYSQL_SERVICE_SERVICE_PORT=3306
TERM=xterm
MYSQL_SERVICE_PORT_3306_TCP_PROTO=tcp
SHLVL=1
KUBERNETES_SERVICE_PORT=443
MYSQL_SERVICE_PORT_3306_TCP_PORT=3306
PROXY_SERVICE_SERVICE_HOST=10.103.187.108
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_SERVICE_HOST=10.96.0.1
MYSQL_SHELL_VERSION=8.0.31-1.el8
ASSIGNMENT3_DJANGO_SERVICE_PORT_8000_TCP_PORT=8000
_=/usr/bin/env
sh-4.4#
```

‘kubectl exec -it mysql-container-ff8844dc9-gs64z /bin/sh’

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3$ kubectl exec -it assignment3-django-deploy-5df896c8f-q5fx9 /bin/sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/GiftcardSite $ echo $SECRET_KEY
kmgysa#fz+9(z1*=c0ydrjizk*7sthm2ga1z4=^61$cxcq8b$l
/GiftcardSite $ env
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.96.0.1:443
HOSTNAME=assignment3-django-deploy-5df896c8f-q5fx9
SECRET_KEY=kmgysa#fz+9(z1*=c0ydrjizk*7sthm2ga1z4=^61$cxcq8b$l
PROXY_SERVICE_SERVICE_HOST=10.103.187.108
PROXY_SERVICE_PORT_8080_TCP_ADDR=10.103.187.108
PYTHON_PIP_VERSION=22.0.4
```

Part 2

In order to run migrations, I created a separate migrationjob.yaml file with environment variables that I want to migrate. Additionally, I made sure the commands “['python3', 'manage.py', 'migrate']” were included which would trigger the django migration job itself via manage.py file

migrationjob.yaml - Visual Studio Code

File Edit Selection View Go Run Terminal Help

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

seedjob.yaml | migrationjob.yaml

home > ubuntu > Documents > appsec > AppSecAssignment3 > part2 > migrationjob.yaml

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: migrationjob
spec:
  template:
    spec:
      containers:
      - name: migrationjob
        image: nyuappsec/assign3:v0
        command: ['python3', 'manage.py', 'migrate']
        env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: securedsecrets
                  key: password

            - name: MYSQL_DB
              value: GiftcardSiteDB

            - name: MYSQL_HOST
              value: mysql-service

            - name: ALLOWED_HOSTS
              value: "*,"

            - name: ADMIN_UNAME
              valueFrom:
                secretKeyRef:
                    name: securedsecrets
                    key: username

            - name: ADMIN_PASS
              valueFrom:
                secretKeyRef:
                    name: securedsecrets
                    key: password

            - name: SECRET_KEY
              valueFrom:
                secretKeyRef:
                    name: securedsecrets
                    key: secret_key

      restartPolicy: Never
  backoffLimit: 7
```

Afterward, I simply ran “kubectl apply -f migrationjob.yaml” which applied the new file and triggered the migration flow.

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$ pwd
/home/ubuntu/Documents/appsec/AppSecAssignment3/part2
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$ kubectl apply -f migrationjob.yaml
job.batch/migrationjob created
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$ kubectl get pods
NAME                                                  READY   STATUS    RESTARTS        AGE
assignment3-django-deploy-5df896c8f-qrxnq             1/1     Running   0               96s
migrationjob-8hlm5                                    1/1     Running   0               5s
mysql-container-ff8844dc9-pp2cf                       1/1     Running   0               98s
prometheus-1669583381-prometheus-node-exporter-kktv5  1/1     Running   2 (2m54s ago)   75m
proxy-89c4bc9c5-8vqhk                                 1/1     Running   0               90s
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$ kubectl get pods
NAME                                                  READY   STATUS      RESTARTS        AGE
assignment3-django-deploy-5df896c8f-qrxnq             1/1     Running     0               114s
migrationjob-8hlm5                                    0/1     Completed   0               23s
mysql-container-ff8844dc9-pp2cf                       1/1     Running     0               116s
prometheus-1669583381-prometheus-node-exporter-kktv5  1/1     Running     2 (3m12s ago)   75m
proxy-89c4bc9c5-8vqhk                                 1/1     Running     0               108s
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$ kubectl get jobs
NAME           COMPLETIONS   DURATION   AGE
migrationjob   1/1           9s         34s
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$
```

I was able to confirm that migration was completed by running kubectl get jobs

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$ kubectl get jobs
NAME           COMPLETIONS   DURATION   AGE
migrationjob   1/1           8s         54s
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$
```

The seeding job took me a while to figure out as it involved populating the records from one table into another. However, similar to the migration job, for the seeding job, I started off with creating a new seedjob.yaml file which would make a call to the database with credentials stored in secrets. Most importantly though, I added the argument to make a call to a separate sql job through the [args] parameter. Furthermore, I made sure to pass the username and password along with the seedjob.sql script so that they can authenticate against secrets that were created in step 1

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: seedjob
spec:
  template:
    spec:
      containers:
      - name: seedjob
        image: 'nyuappsec/assign3-db:v0'
        command: ["/bin/sh"]
        args: ["-c","mysql --user=root --password=${MYSQL_ROOT_PASSWORD} --database=${MYSQL_DATABASE} --host=mysql-service -f < /docker-entrypoint-initdb.d/seedjob.sql"]
        env:
          - name: MYSQL_ROOT_PASSWORD
            valueFrom:
              secretKeyRef:
                name: securedsecrets
                key: password

          - name:  MYSQL_DATABASE
            value: GiftcardSiteDB

      restartPolicy: Never
  backoffLimit: 3
```

As to avoid duplication in records being added, I simply adjusted the docker file contained in DB folder to comment out the data folder and initial setup.sql command as this was already run on setup.

```
Open                                   Dockerfile
                                       ~/Documents/appsec/AppSecAssignment3/part2
FROM mysql:latest

#RUN mkdir /data
COPY ./products.csv /products.csv
COPY ./users.csv /users.csv
#COPY ./setup.sql /docker-entrypoint-initdb.d/setup.sql
COPY ./seedingjob.sql /docker-entrypoint-initdb.d/seedingjob.sql

ENTRYPOINT ["/entrypoint.sh"]
CMD ["mysqld", "--secure-file-priv=/"]
#CMD ["--secure-file-priv=/"]
```

In continuity to earlier step, I adjusted the setup.sql file to comment out the extra data load mentioned here as to avoid duplication with the earlier job.

```
--
--COMMENTING THESE OUT AS TO AVOID DUPLICATION WITH SEEDJOB.SQL
---LOAD DATA INFILE '/data/products.csv' INTO TABLE LegacySite_product FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
LINES TERMINATED BY '\r\n';
--
-- Put user into table.
--
---LOAD DATA INFILE '/data/users.csv' INTO TABLE LegacySite_user FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"' LINES
TERMINATED BY '\r\n';
SQL   Tab Width: 8   Ln 123, Col 8   INS
```

As we commented out these lines, I simply re-added them those lines ‘separately’ into the file ‘seedjob.sql’ which would run along with username and password arguments when seedjob.yaml file is applied

```
Open                                   seedjob.sql                                   Save
                                       ~/Documents/appsec/AppSecAssignment3/part2
secrets.txt   seedjob_-.yaml   Dockerfile   setup.sql   seedjob.sql
LOAD DATA INFILE '/products.csv' INTO TABLE LegacySite_product FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"' LINES
TERMINATED BY '\r\n';
LOAD DATA INFILE '/users.csv' INTO TABLE LegacySite_user FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY
'\r\n';
```

Finally, similarly to the migration job, I ran “kubectl apply -f seedjob.yaml” to trigger the seeding job.

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$ kubectl apply -f seedjob.yaml
job.batch/seedjob created
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$ kubectl get jobs
NAME           COMPLETIONS   DURATION   AGE
migrationjob   1/1           7s         101s
seedjob        0/1           3s         3s
```

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$ kubectl get jobs
NAME           COMPLETIONS   DURATION   AGE
migrationjob   1/1           7s         107s
seedjob        1/1           7s         9s
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part2$ kubectl get pods
NAME                                        READY   STATUS      RESTARTS   AGE
assignment3-django-deploy-5df896c8f-bzvcp   1/1     Running     0          9m7s
migrationjob-cdwjd                          0/1     Completed   0          114s
mysql-container-7887b7f64b-6vx4k            1/1     Running     0          10m
proxy-89c4bc9c5-xzdvn                       1/1     Running     0          10m
seedjob-w5h57                               0/1     Completed   0          16s
```

The confirm completion of seedjob, I ran kubectl get jobs which showed both migrationjob and seedjob complete

References:
https://stackoverflow.com/questions/51577441/how-to-seed-django-project-insert-a-bunch-of-data-into-the-project-for-initi
https://docs.djangoproject.com/en/4.1/howto/initial-data/
https://stackoverflow.com/questions/65974627/clearing-and-seeding-database-from-endpoint-in-django
https://medium.com/@ardho/migration-and-seeding-in-django-3ae322952111
https://stackoverflow.com/questions/60141107/how-do-i-run-django-seed-data-in-my-mysql-docker-image
https://stackoverflow.com/questions/59940160/is-there-a-way-in-a-seed-data-yaml-file-to-autogenerate-models-on-which-first-mo
https://github.com/Brobin/django-seed
https://www.coursehero.com/tutors-problems/Computer-Science/30588605-How-to-create-a-Kubernetes-migrations-job-using-a-YAML-file-for-a-MySQ/

Part 3.1

For Part 3.1, I started by analyzing with exploring prometheus security best practices with reading articles (https://jfrog.com/dont-let-prometheus-steal-your-fire/ and https://danuka-praneeth.medium.com/setting-up-a-comprehensive-monitoring-system-with-prometheus-6b2b73cf54d2). Following that, I carefully analyzed the views.py file for exposure of any sensitive metrics. The most obvious exposure was related 'graphs[pword].inc()' which logged secret use during a POST request. I initially noticed this in part 1 as well and commented a portion of it out, but full remediated it now by commenting out everything from "if pword.. " to "graphs[pword].inc()".

```
        # KG: Uh... I'm not sure this makes sense.
        # Collect data to ensure good password use.
        #PART 3 - FIX: commenting this out as this may reveal secrets during login process in debug logs
        ## recording the password in counter is a huge issue and is remediated by commenting out this entirely

        #if pword not in graphs.keys():
        #    graphs[pword] = Counter(f'counter_{pword}', 'The total number of '\
        #      + f'times {pword} was used')
        #graphs[pword].inc()
```

Additionally, I commented out the tracker for the number of login requests as this may reveal sensitive behavior from customers

```
# Prometheus stuff!
graphs = {}
graphs['r_counter'] = Counter('python_request_r_posts', 'The total number'\
  + ' of register posts.')
#graphs['l_counter'] = Counter('python_request_l_posts', 'The total number'\
#  + ' of login posts.')         #PART 3 FIX: graphs['l_counter'].inc() we don't want to expose number of logins
graphs['b_counter'] = Counter('python_request_b_posts', 'The total number'\
```

Part 3.2

For Part 3.2, I decided to expand monitoring by also logging the 404 errors returned. Accordingly, I added the ‘error_return_counter’ graph which would track the 404 errors on return statements

```
 + ' of card use posts.')
##PART 3 - adding logging for error messages
graphs['error_return_counter'] = Counter('python_return_error', 'The total number'\
 + ' of errors returned')

```

Next, I searched for ‘404 Not found’ across the views.py file to find instances where an error is returned and found 4 of such instances. Accordingly, I added the graphs['error_return_counter'].inc() statement to each instance of error return, which would respectively increment the counter with each error in db.

```
def buy_card_view(request, prod_num=0):
    if request.method == 'GET':
        context = {"prod_num" : prod_num}
        director = request.GET.get('director', None)
        if director is not None:
            # KG: Wait, what is this used for? Need to check the template.
            context['director'] = director
        if prod_num != 0:
            try:
                prod = Product.objects.get(product_id=prod_num)
            except:
                return HttpResponse("ERROR: 404 Not Found.")
                graphs['error_return_counter'].inc() ##Part 3 - track error returns
        else:
            try:
                prod = Product.objects.get(product_id=1)
            except:
                return HttpResponse("ERROR: 404 Not Found.")
                graphs['error_return_counter'].inc() ##Part 3 - track error returns
```

```
                prod = Product.objects.get(product_id=prod_num)
            except:
                return HttpResponse("ERROR: 404 Not Found.")
                graphs['error_return_counter'].inc() ##Part 3 - track error returns
        else:
            try:
                prod = Product.objects.get(product_id=1)
            except:
                return HttpResponse("ERROR: 404 Not Found.")
                graphs['error_return_counter'].inc() ##Part 3 - track error returns
        context['prod_name'] = prod.product_name
```

Part 3.3

I started off by downloading helm using the instructions on its respective website (https://helm.sh/docs/intro/install/)

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ chmod 700 get_helm.sh
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.10.2-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$
```

Next, using the recently installed helm repo, I installed prometheus from its respective repo referenced here (https://github.com/prometheus-community/helm-charts)

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ helm install prometheus-community/prometheus --generate-name
NAME: prometheus-1669583381
LAST DEPLOYED: Sun Nov 27 16:09:42 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-1669583381-server.default.svc.cluster.local


Get the Prometheus server URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=prometheus,component=server" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace default port-forward $POD_NAME 9090


The Prometheus alertmanager can be accessed via port  on the following DNS name from within your cluster:
prometheus-1669583381-%!s(<nil>).default.svc.cluster.local


Get the Alertmanager URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=prometheus,component=" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace default port-forward $POD_NAME 9093
#################################################################################
######   WARNING: Pod Security Policy has been disabled by default since    #####
######            it deprecated after k8s 1.25+. use                        #####
######            (index .Values "prometheus-node-exporter" "rbac"          #####
###### .          "pspEnabled") with (index .Values                         #####
######            "prometheus-node-exporter" "rbac" "pspAnnotations")       #####
######            in case you still need it.                                #####
#################################################################################


The Prometheus PushGateway can be accessed via port 9091 on the following DNS name from within your cluster:
prometheus-1669583381-prometheus-pushgateway.default.svc.cluster.local


Get the PushGateway URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=prometheus-pushgateway,component=pushgateway" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace default port-forward $POD_NAME 9091

For more information on running Prometheus, visit:
https://prometheus.io/
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$
```

In accordance with instructions in tutorial (https://artifacthub.io/packages/helm/prometheus-community/prometheus), I ran kubectl get services and kubectl get pods to ensure prometheus was running accordingly

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ kubectl get services
NAME                                            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
assignment3-django-service                      NodePort    10.104.70.181    <none>        8000:31219/TCP   16d
kubernetes                                      ClusterIP   10.96.0.1        <none>        443/TCP          16d
mysql-service                                   ClusterIP   10.102.210.234   <none>        3306/TCP         16d
prometheus-1669583381-alertmanager              ClusterIP   10.103.41.227    <none>        9093/TCP         62s
prometheus-1669583381-alertmanager-headless     ClusterIP   None             <none>        9093/TCP         62s
prometheus-1669583381-kube-state-metrics        ClusterIP   10.98.198.6      <none>        8080/TCP         62s
prometheus-1669583381-prometheus-node-exporter  ClusterIP   10.96.82.154     <none>        9100/TCP         62s
prometheus-1669583381-prometheus-pushgateway    ClusterIP   10.109.153.6     <none>        9091/TCP         62s
prometheus-1669583381-server                    ClusterIP   10.97.39.86      <none>        80/TCP           62s
proxy-service                                   NodePort    10.103.187.108   <none>        8080:31519/TCP   16d
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ kubectl get pods
NAME                                                READY   STATUS    RESTARTS       AGE
assignment3-django-deploy-5df896c8f-q5fx9           1/1     Running   1 (135m ago)   137m
mysql-container-6597c45f8-2tgf7                     1/1     Running   1 (135m ago)   137m
prometheus-1669583381-alertmanager-0                2/2     Running   0              113s
prometheus-1669583381-kube-state-metrics-58d965777c-xghwt   1/1     Running   0              113s
prometheus-1669583381-prometheus-node-exporter-xtcxn        1/1     Running   0              113s
prometheus-1669583381-prometheus-pushgateway-7f69c4f7b5-tlppp   1/1     Running   0              113s
prometheus-1669583381-server-88d7b9746-sljsf        2/2     Running   0              113s
proxy-89c4bc9c5-gfplt                               1/1     Running   4 (133m ago)   143m
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$
```

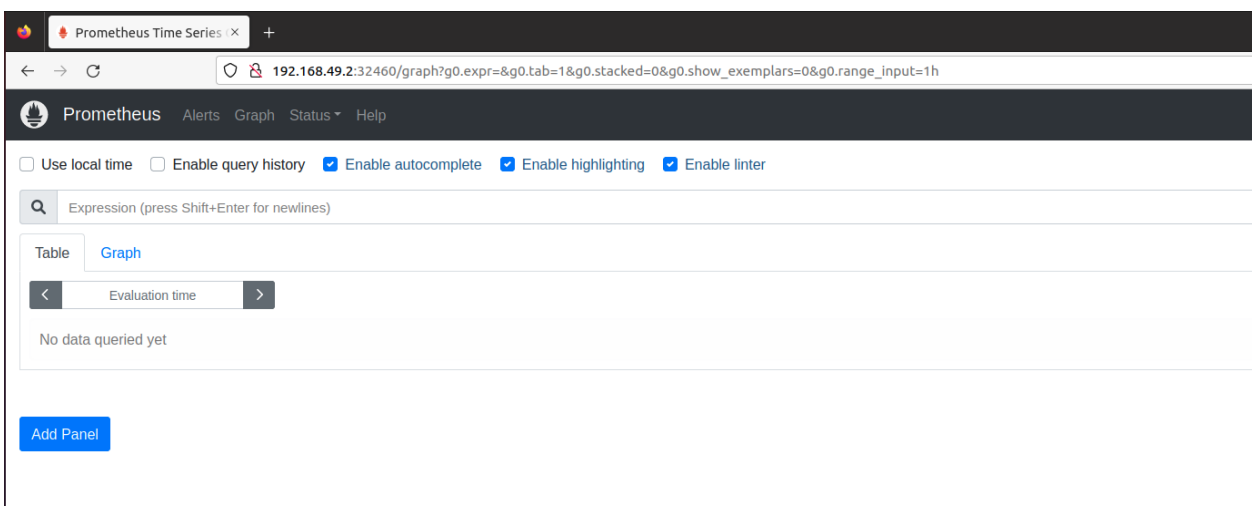
Next, I ran ‘minikube service --all’ to locate all services running and isolate the prometheus server

```
😿  service default/prometheus-1669583381-prometheus-pushgateway has no node port
|-----------|-----------------------------|-------------|--------------|
| NAMESPACE |             NAME            | TARGET PORT |     URL      |
|-----------|-----------------------------|-------------|--------------|
| default   | prometheus-1669583381-server |             | No node port |
|-----------|-----------------------------|-------------|--------------|
```

From there, I ran ‘kubectl expose service prometheus-1669583381-server --type=NodePort --target-port=9090 --name=prometheus-server-np’ to expose the appropriate prometheus server to its respective port

```
|-----------|----------------------|-------------|---------------------------|
| NAMESPACE |         NAME         | TARGET PORT |            URL            |
|-----------|----------------------|-------------|---------------------------|
| default   | prometheus-server-np |          80 | http://192.168.49.2:32460 |
|-----------|----------------------|-------------|---------------------------|
|-----------|---------------|-------------|---------------------------|
| NAMESPACE |     NAME      | TARGET PORT |            URL            |
|-----------|---------------|-------------|---------------------------|
| default   | proxy-service |        8080 | http://192.168.49.2:31519 |
|-----------|---------------|-------------|---------------------------|
🎉  Opening service default/assignment3-django-service in default browser...
🎉  Opening service default/prometheus-server-np in default browser...
🎉  Opening service default/proxy-service in default browser...
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$
```

Additionally, I was able to confirm that it was running by visiting its respective webpage for prometheus server which is running (http://192.168.49.2:32460/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.show_exemplars=0&g0.range_input=1h)



Next, I had to adjust the configmap so the prometheus server would be to pull metrics from our website. I did this by first running ‘kubectl get configmap’ to get the list of configmaps avaliable.

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ kubectl get configmap
NAME                                   DATA   AGE
kube-root-ca.crt                       1      17d
prometheus-1669583381-alertmanager     1      14m
prometheus-1669583381-server           6      14m
```

Next, I edited the configmap by running ‘kubectl edit configmap prometheus-1669583381-server’ for prometheus server to point to another job for GiftCardSite_Monitoring as follows:

```
ubuntu@ubuntu2004: ~/Documents/appsec/AppSecAssignment3/part3
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  alerting_rules.yml: |
    {}
  alerts: |
    {}
  allow-snippet-annotations: "false"
  prometheus.yml: |
    global:
      evaluation_interval: 1m
      scrape_interval: 20s
      scrape_timeout: 10s
    rule_files:
    - /etc/config/recording_rules.yml
    - /etc/config/alerting_rules.yml
    - /etc/config/rules
    - /etc/config/alerts
    scrape_configs:
    - job_name: prometheus
      scrape_interval: 10s
      scrape_timeout: 15s
      static_configs:
      - targets:
        - localhost:9090
      ## Part 3.3 adding monitoring for giftcardsite
    - job_name: GiftCardSite_monitoring
      static_configs:
      - targets:
        - proxy-service:8080
    - bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      job_name: kubernetes-apiservers
      kubernetes_sd_configs:
      - role: endpoints
      relabel_configs:
      - action: keep
        regex: default;kubernetes;https
        source_labels:
        - __meta_kubernetes_namespace
        - __meta_kubernetes_service_name
        - __meta_kubernetes_endpoint_port_name
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
    - bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      job_name: kubernetes-nodes
      kubernetes_sd_configs:
```

Once edited, it was confirmed that the config was saved with the following response:

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ kubectl edit configmap prometheus-16695
83381-server
configmap/prometheus-1669583381-server edited
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$
```

Lastly, I piped a copy of the updated YAML file by running
kubectl get configmap prometheus-1669583381-server -o yaml >
promethus-server_updated_copy.yaml

Following this change, I simply restarted the PODS (by deleting each one via kubectl delete pod <pod_name>) and confirmed they were running accordingly after (kubectl get pods):

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ kubectl get pods
NAME                                                    READY   STATUS    RESTARTS   AGE
assignment3-django-deploy-5df896c8f-pzcml               1/1     Running   0          7m57s
mysql-container-6597c45f8-gsk7p                         1/1     Running   0          7m10s
prometheus-1669583381-alertmanager-0                    2/2     Running   0          6m39s
prometheus-1669583381-kube-state-metrics-58d965777c-kt8w5   1/1     Running   0          6m32s
prometheus-1669583381-prometheus-node-exporter-4c8p2    1/1     Running   0          6m20s
prometheus-1669583381-prometheus-pushgateway-7f69c4f7b5-lpnxt   1/1     Running   0          6m12s
prometheus-1669583381-server-88d7b9746-5znw8            1/2     Running   0          11s
proxy-89c4bc9c5-t6q4c                                   1/1     Running   0          5m42s
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ minikube service list
|-------------|-----------------------------------------------|--------------|---------------------------|
|  NAMESPACE  |                     NAME                      | TARGET PORT  |            URL            |
|-------------|-----------------------------------------------|--------------|---------------------------|
| default     | assignment3-django-service                    |         8000 | http://192.168.49.2:31219 |
| default     | kubernetes                                    | No node port |
| default     | mysql-service                                 | No node port |
| default     | prometheus-1669583381-alertmanager            | No node port |
| default     | prometheus-1669583381-alertmanager-headless   | No node port |
| default     | prometheus-1669583381-kube-state-metrics      | No node port |
| default     | prometheus-1669583381-prometheus-node-exporter | No node port |
| default     | prometheus-1669583381-prometheus-pushgateway  | No node port |
| default     | prometheus-1669583381-server                  | No node port |
| default     | prometheus-server-np                          |           80 | http://192.168.49.2:32460 |
| default     | proxy-service                                 |         8080 | http://192.168.49.2:31519 |
| kube-system | kube-dns                                      | No node port |
|-------------|-----------------------------------------------|--------------|---------------------------|
```

Additionally, I went into the /etc directory in server separately by running kubectl exec --it prometheus-1669583381-server-88d7b9746-5znw8 /bin/sh to ensure the YAML file updated accordingly

Finally, I ran “minkube service list prometheus” to confirm that the server was pointing to a PORT.

```
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$ minikube service list prometheus
|-------------|-----------------------------------------------|--------------|---------------------------|
|  NAMESPACE  |                     NAME                      | TARGET PORT  |            URL            |
|-------------|-----------------------------------------------|--------------|---------------------------|
| default     | assignment3-django-service                    |         8000 | http://192.168.49.2:31219 |
| default     | kubernetes                                    | No node port |
| default     | mysql-service                                 | No node port |
| default     | prometheus-1669583381-alertmanager            | No node port |
| default     | prometheus-1669583381-alertmanager-headless   | No node port |
| default     | prometheus-1669583381-kube-state-metrics      | No node port |
| default     | prometheus-1669583381-prometheus-node-exporter | No node port |
| default     | prometheus-1669583381-prometheus-pushgateway  | No node port |
| default     | prometheus-1669583381-server                  | No node port |
| default     | prometheus-server-np                          |           80 | http://192.168.49.2:32460 |
| default     | proxy-service                                 |         8080 | http://192.168.49.2:31519 |
| kube-system | kube-dns                                      | No node port |
|-------------|-----------------------------------------------|--------------|---------------------------|
ubuntu@ubuntu2004:~/Documents/appsec/AppSecAssignment3/part3$
```

As a confirmation check for myself, I visited the website and confirmed that monitoring for giftcard site was present

Prometheus Time Series

192.168.49.2:32460/service-discovery?search=

Prometheus Alerts Graph Status Help

Service Discovery

Filter by labels

- GiftCardSite_monitoring (1 / 1 active targets)
- kubernetes-apiservers (1 / 16 active targets)
- kubernetes-nodes (1 / 1 active targets)
- kubernetes-nodes-cadvisor (1 / 1 active targets)
- kubernetes-service-endpoints (3 / 14 active targets)
- prometheus (1 / 1 active targets)
- prometheus-pushgateway (1 / 14 active targets)
- kubernetes-pods (0 / 21 active targets)
- kubernetes-pods-slow (0 / 21 active targets)
- kubernetes-service-endpoints-slow (0 / 16 active targets)
- kubernetes-services (0 / 14 active targets)

GiftCardSite_monitoring

show less

Discovered Labels	Target Labels
__address__="proxy-service:8080" __metrics_path__="/metrics" __scheme__="http" __scrape_interval__="20s" __scrape_timeout__="10s" job="GiftCardSite_monitoring"	instance="proxy-service:8080" job="GiftCardSite_monitoring"

kubernetes-apiservers

show more

Prometheus Time Series

192.168.49.2:32460/targets?search=

Prometheus Alerts Graph Status Help

GiftCardSite_monitoring (1/1 up)

show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://proxy-service:8080/metrics	UP	instance="proxy-service:8080" job="GiftCardSite_monitoring"	2.904s ago	35.675ms	

kubernetes-apiservers (1/1 up)

show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.49.2:8443/metrics	UP	instance="192.168.49.2:8443" job="kubernetes-apiservers"	21.664s ago	234.342ms	

kubernetes-nodes (1/1 up)

show less

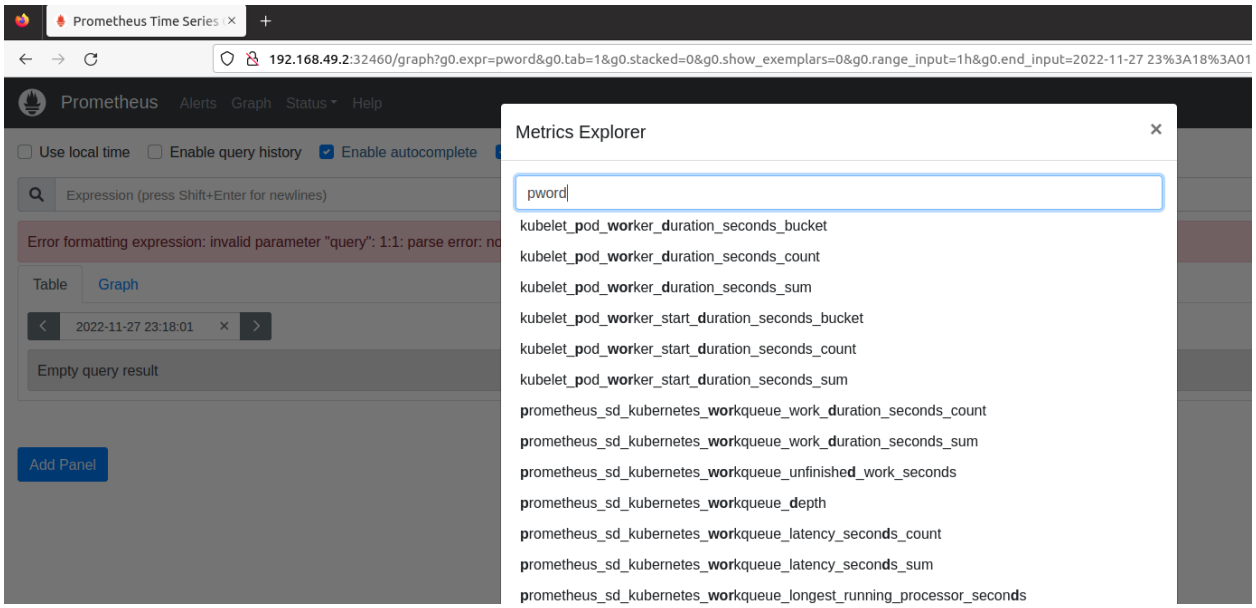
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://kubernetes.default.svc/api/v1/nodes/minikube/proxy/metrics	UP	beta_kubernetes_io_arch="amd64" beta_kubernetes_io_os="linux" instance="minikube" job="kubernetes-nodes" kubernetes_io_arch="amd64" kubernetes_io_hostname="minikube" kubernetes_io_os="linux" minikube_k8s_io_commit="986b1ebd987211ed16f8cc10aed7d2c42fc8392f" minikube_k8s_io_name="minikube" minikube_k8s_io_primary="true" minikube_k8s_io_updated_at="2022_11_10T16_14_22_0700" minikube_k8s_io_version="v1.28.0"	11.408s ago	3.807s	

kubernetes-nodes-cadvisor (1/1 up)

show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://kubernetes.default.svc/api/v1/nodes/minikube/proxy/metrics/cadvisor	UP	beta_kubernetes_io_arch="amd64" beta_kubernetes_io_os="linux" instance="minikube" job="kubernetes-nodes-cadvisor" kubernetes_io_arch="amd64" kubernetes_io_hostname="minikube"	18.160s ago	724.065ms	

Additionally, I checked to make sure that the counter ‘pword’ filter didn’t exist and confirmed it was not present which means views.py is reflecting everything accordingly!



References used:

https://prometheus.io/docs/prometheus/latest/configuration/configuration/
https://prometheus.io/docs/introduction/first_steps/
https://www.redhat.com/sysadmin/installing-prometheus
https://github.com/bakins/minikube-prometheus-demo
https://docs.timescale.com/timescaledb/latest/tutorials/monitor-django-with-prometheus/
https://grafana.com/grafana/dashboards/9528-django-prometheus/