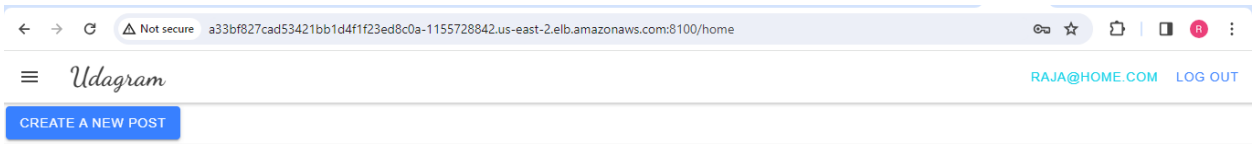
Final Project – Refactor Udagram App into Microservices and Deploy

The source code for the project is at https://github.com/rraerrabotu/udagram-refactor-project

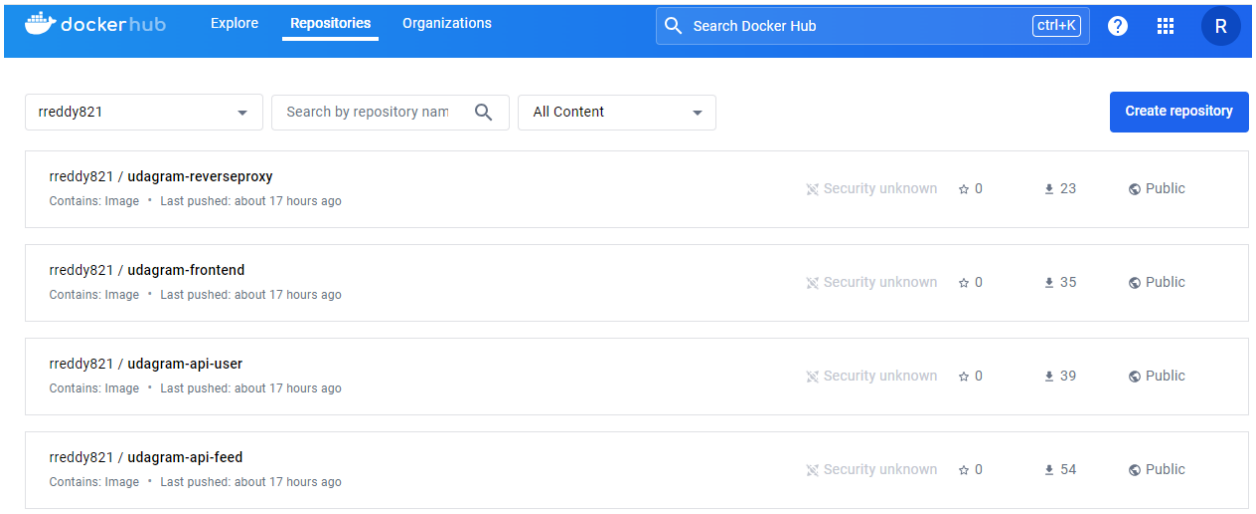
Screenshot of Udagram deployed in EKS with External IP exposed



Containers and Microservices

Success Criteria	Specifications	Result
Divide an application into microservices	/feed and /user backends are separated into independent projects.	***Completed –*** *Please look at GitHub*
Build and run a container image using Docker	Project includes Dockerfiles to successfully create Docker images for /feed, /user backends, project frontend, and reverse proxy. Screenshot of DockerHub shows the images.	***Completed*** 1) *Please look at GitHub* 2) *DockerHub screen shot attached below*

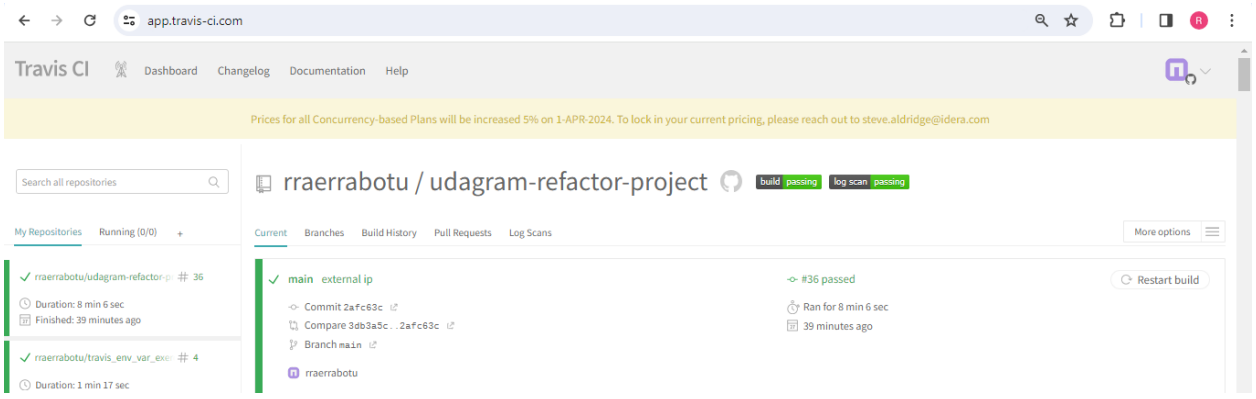
Screenshot of DockerHub



Independent Releases and Deployments

Success Criteria	Specifications	Result
Divide an application into microservices	Project includes a .travis.yml file. Screenshot of the Travis CI interface shows a successful build and deploy job.	***Completed –*** 1) *Please look at GitHub for .travis.yaml* 2) *Screenshot of Travis CI*

Screenshot of the Travis CI interface



Service Orchestration with Kubernetes

Success Criteria	Specifications	Result
Deploy microservices using a Kubernetes cluster on AWS	A screenshots of kubectl commands show the Frontend and API projects deployed in Kubernetes. The output of kubectl get pods indicates that the pods are running successfully with the STATUS value Running. The output of kubectl describe services does not expose any sensitive strings such as database passwords.	***Completed – Screenshots below***

Screenshot of the "kubectl"commands

```
topgun@LAPTOP-IO9H2UDN MINGW64 ~/Course3-Exercises/refactor-udagram/udagram-eks (main)
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS      AGE
backend-feed-7688fbd9d7-4rjxt       1/1     Running   0             70m
backend-feed-7688fbd9d7-94bzt       1/1     Running   0             70m
backend-feed-7688fbd9d7-gcs6b       1/1     Running   0             70m
backend-user-66c6467b59-bjrlb       1/1     Running   0             70m
backend-user-66c6467b59-l7nsg       1/1     Running   0             70m
frontend-7cdfbb7f88-fsdn7           1/1     Running   0             44m
reverseproxy-6589fd69b6-rsb7h       1/1     Running   2 (70m ago)   70m
```

```
topgun@LAPTOP-IO9H2UDN MINGW64 ~/Course3-Exercises/refactor-udagram/udagram-eks (main)
$ kubectl get services
NAME                 TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)          AGE
backend-feed         ClusterIP      10.100.105.52    <none>                                                                    8080/TCP         70m
backend-user         ClusterIP      10.100.5.120     <none>                                                                    8080/TCP         70m
kubernetes           ClusterIP      10.100.0.1       <none>                                                                    443/TCP          82m
publicfrontend       LoadBalancer   10.100.56.171    a33bf827cad53421bb1d4f1f23ed8c0a-1155728842.us-east-2.elb.amazonaws.com   8100:32752/TCP   43m
publicreverseproxy   LoadBalancer   10.100.55.128    a9981d9a5c6c54740be6edcff4b46932-1956909875.us-east-2.elb.amazonaws.com   8080:31993/TCP   65m
reverseproxy         ClusterIP      10.100.208.36    <none>                                                                    8080/TCP         70m
```

```
topgun@LAPTOP-IO9H2UDN MINGW64 ~/Course3-Exercises/refactor-udagram/udagram-eks (main)
$ kubectl get deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
backend-feed   3/3     3            3           72m
backend-user   2/2     2            2           72m
frontend       1/1     1            1           72m
reverseproxy   1/1     1            1           71m
```

kubectl_describe_services output - *https://github.com/rraerrabotu/udagram-refactor-project/blob/main/udagram-finalproject-submission/kubectl_describe-services-output.txt*

Success Criteria	Specifications	Result
Use a reverse proxy to direct requests to the appropriate backend	Screenshot of Kubernetes services shows a reverse proxy	***Completed – Screenshot below***

```
topgun@LAPTOP-IO9H2UDN MINGW64 ~/Course3-Exercises/refactor-udagram/udagram-eks (main)
$ kubectl describe service reverseproxy
Name:              reverseproxy
Namespace:         default
Labels:            service=reverseproxy
Annotations:       <none>
Selector:          service=reverseproxy
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.100.208.36
IPs:               10.100.208.36
Port:              8080  8080/TCP
TargetPort:        8080/TCP
Endpoints:         192.168.31.136:8080
Session Affinity:  None
Events:            <none>
```

Success Criteria	Specifications	Result
Configure scaling and self-healing for each service	Kubernetes services are replicated. At least one of the Kubernetes services has replicas: defined with a value greater than 1 in its deployment.yml file. Screenshot of Kubernetes cluster of command kubectl describe hpa has autoscaling configured with CPU metrics.	***Completed – Screenshot below***

Showing Replicas

backend-feed-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    service: backend-feed
  name: backend-feed
spec:
  replicas: 3
  selector:
    matchLabels:
      service: backend-feed
  template:
    metadata:
      labels:
        service: backend-feed
    spec:
      containers:
      - image: rreddy821/udagram-api-feed:latest
        name: backend-feed
        imagePullPolicy: Always
        ports:
        - containerPort: 8080
```

HPA Setup for backend-feed

```
topgun@LAPTOP-IO9H2UDN MINGW64 ~/Course3-Exercises/refactor-udagram/udagram-eks (main)
$ kubectl autoscale deployment backend-feed --cpu-percent=50 --min=3 --max=5
horizontalpodautoscaler.autoscaling/backend-feed autoscaled

topgun@LAPTOP-IO9H2UDN MINGW64 ~/Course3-Exercises/refactor-udagram/udagram-eks (main)
$ kubectl get hpa
NAME           REFERENCE                 TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
backend-feed   Deployment/backend-feed   <unknown>/50%   3         5         3          19s

topgun@LAPTOP-IO9H2UDN MINGW64 ~/Course3-Exercises/refactor-udagram/udagram-eks (main)
$ kubectl describe hpa
Name:                                                  backend-feed
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Wed, 20 Mar 2024 11:56:29 -0500
Reference:                                             Deployment/backend-feed
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  <unknown> / 50%
Min replicas:                                          3
Max replicas:                                          5
Deployment pods:                                       3 current / 0 desired
Conditions:
  Type           Status  Reason                   Message
  ----           ------  ------                   -------
  AbleToScale    True    SucceededGetScale        the HPA controller was able to get the target's current scale
  ScalingActive  False   FailedGetResourceMetric  the HPA was unable to compute the replica count: failed to get cpu utilization: unable to get metrics for resource cp
u: unable to fetch metrics from resource metrics API: the server could not find the requested resource (get pods.metrics.k8s.io)
Events:
  Type     Reason                        Age                From                       Message
  ----     ------                        ----               ----                       -------
  Warning  FailedGetResourceMetric       4s (x6 over 79s)   horizontal-pod-autoscaler  failed to get cpu utilization: unable to get metrics for resource cpu: unable to
fetch metrics from resource metrics API: the server could not find the requested resource (get pods.metrics.k8s.io)
  Warning  FailedComputeMetricsReplicas  4s (x6 over 79s)   horizontal-pod-autoscaler  invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilizati
on: unable to get metrics for resource cpu: unable to fetch metrics from resource metrics API: the server could not find the requested resource (get pods.metrics.k8s.i
o)
```

Debugging, Monitoring, and Logging

Success Criteria	Specifications	Result
Use logs to capture metrics for debugging a microservices deployment	Screenshot of one of the backend API pod logs indicates user activity that is logged when an API call is made.	***Completed – Screenshot below***