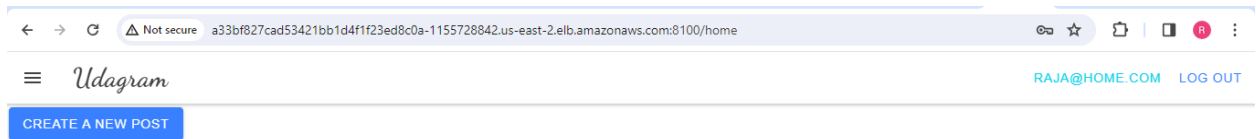


Final Project – Refactor Udagram App into Microservices and Deploy

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

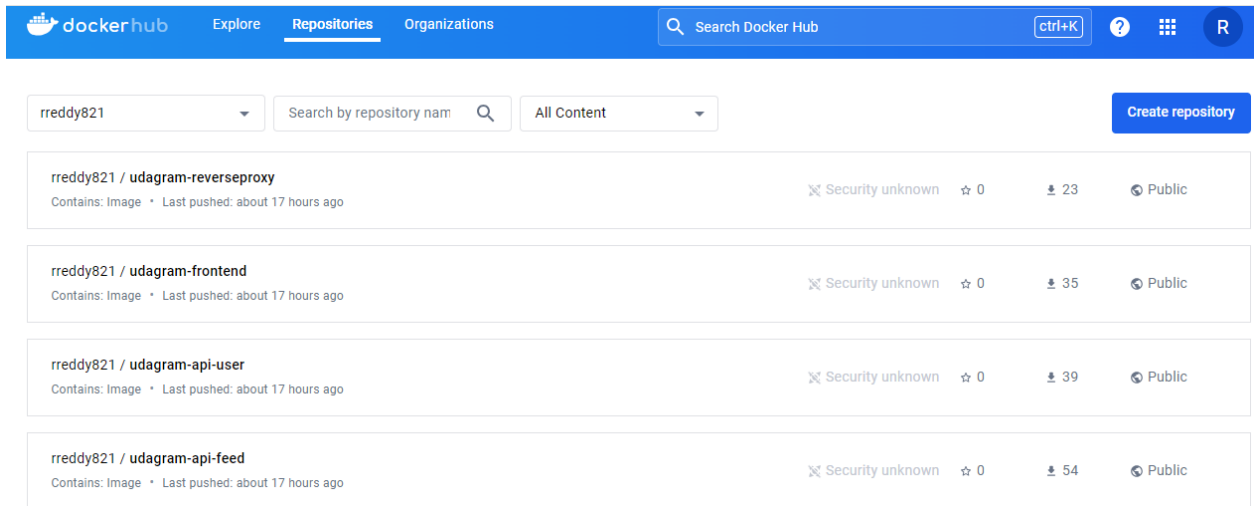
Screenshot of Udagram deployed in EKS with External IP exposed



Containers and Microservices

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

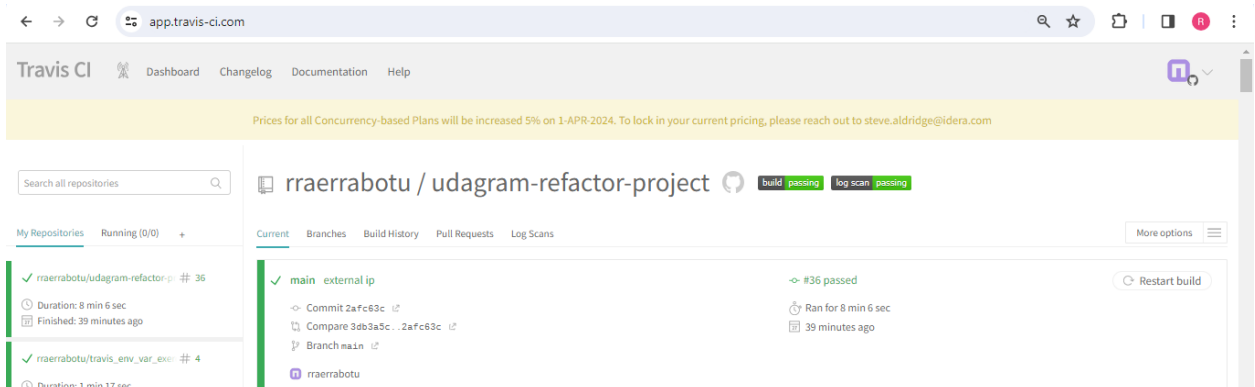
Screenshot of DockerHub



Independent Releases and Deployments

Success Criteria	Specifications	Result
Divide an application into microservices	<p>Project includes a .travis.yml file.</p> <p>Screenshot of the Travis CI interface shows a successful build and deploy job.</p>	<p>Completed –</p> <ol style="list-style-type: none">1) Please look at GitHub for .travis.yml2) 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	<p>A screenshots of <code>kubectl</code> commands show the Frontend and API projects deployed in Kubernetes.</p> <p>The output of <code>kubectl get pods</code> indicates that the pods are running successfully with the <code>STATUS</code> value <code>Running</code>.</p> <p>The output of <code>kubectl describe services</code> does not expose any sensitive strings such as database passwords.</p>	Completed – Screenshots below

Screenshot of the “kubectl” commands

```
topgun@LAPTOP-I09H2UDN 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-bjr1b       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-I09H2UDN 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  a9981d9a5c6c54740be6edc4b46932-1956909875.us-east-2.elb.amazonaws.com  8080:31993/TCP  65m
reverseproxy        ClusterIP   10.100.208.36  <none>         8080/TCP   70m
```

```
topgun@LAPTOP-I09H2UDN 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
```

kubect!_describe_services output - https://github.com/rraerrabotu/udagram-refactor-project/blob/main/udagram-finalproject-submission/kubect!_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-I09H2UDN 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	<p>Kubernetes services are replicated. At least one of the Kubernetes services has <code>replicas:</code> defined with a value greater than 1 in its <code>deployment.yml</code> file.</p> <p>Screenshot of Kubernetes cluster of command <code>kubectl describe hpa</code> has autoscaling configured with CPU metrics.</p>	Completed – Screenshot below

Showing Replicas

```
backend-feed-deployment.yml x
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      service: backend-feed
6    name: backend-feed
7  spec:
8    replicas: 3
9    selector:
10     matchLabels:
11       service: backend-feed
12   template:
13     metadata:
14       labels:
15         service: backend-feed
16     spec:
17       containers:
18       - image: rreddy821/udagram-api-feed:latest
19         name: backend-feed
20         imagePullPolicy: Always
21         ports:
22         - containerPort: 8080
```

HPA Setup for backend-feed – Updated based on review feedback

```
topgun@LAPTOP-I09H2UDN MINGW64 ~/Course3-Exercises/refactor-udagram/udagram-eks (main)
$ kubectl get hpa
NAME                REFERENCE                TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
backend-feed        Deployment/backend-feed   0%/50%   3         5         3          2m53s

topgun@LAPTOP-I09H2UDN 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 14:37:20 -0500
Reference:           Deployment/backend-feed
Metrics:              ( current / target )
  resource cpu on pods (as a percentage of request): 0% (0) / 50%
Min replicas:        3
Max replicas:        5
Deployment pods:      3 current / 3 desired
Conditions:
  Type      Status  Reason                        Message
  ----      -
  AbleToScale  True    ScaledDownStabilized         recent recommendations were higher than current one, applying the highest recent recommendation
  ScalingActive  True    ValidMetricFound              the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited False    DesiredWithinRange            the desired count is within the acceptable range
Events:      <none>
```

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

```
topgun@LAPTOP-I09H2UDN MINGW64 ~/Course3-Exercises/refactor-udagram (main)
$ kubectl logs backend-user-66c6467b59-bjrlb

> udagram-api@2.0.0 prod /usr/src/app
> ts-node-dev --respawn --transpile-only ./src/server.ts

[INFO] 15:23:06 ts-node-dev ver. 1.1.8 (using ts-node ver. 9.1.1, typescript ver. 3.9.10)
Initialize database connection...
UserName: postgres
Password: Anvesh123
database: postgres
Host: udagram-ms-project-db-1.cstyeuhjxn6n.us-east-1.rds.amazonaws.com
URL: http://localhost:8100
Executing (default): CREATE TABLE IF NOT EXISTS "User" ("email" VARCHAR(255), "passwordHash" VARCHAR(255), "createdAt" TIMESTAMP WITH TIME ZONE, "updatedAt" TIMESTAMP WITH TIME ZONE, PRIMARY KEY ("email"));
Executing (default): SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey, array_agg(a.attnum) as column_indexes, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM pg_class t, pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indexrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND t.relkind = 'r' and t.relname = 'User' GROUP BY i.relname, ix.indexrelid, ix.indkey ORDER BY i.relname;
Listening on port is: 8080
2. Listening on port is: 8080
server running http://localhost:8100
press CTRL+C to stop server
Executing (default): SELECT "email", "passwordHash", "createdAt", "updatedAt" FROM "User" AS "User" WHERE "User"."email" = 'raj@home.com';
Executing (default): INSERT INTO "User" ("email","passwordHash","createdAt","updatedAt") VALUES ($1,$2,$3,$4) RETURNING *;
Executing (default): SELECT "email", "passwordHash", "createdAt", "updatedAt" FROM "User" AS "User" WHERE "User"."email" = 'raj@home.com';
Executing (default): SELECT "email", "passwordHash", "createdAt", "updatedAt" FROM "User" AS "User" WHERE "User"."email" = 'home@home.com';
Executing (default): INSERT INTO "User" ("email","passwordHash","createdAt","updatedAt") VALUES ($1,$2,$3,$4) RETURNING *;
Executing (default): SELECT "email", "passwordHash", "createdAt", "updatedAt" FROM "User" AS "User" WHERE "User"."email" = 'raja@home.com';
Executing (default): INSERT INTO "User" ("email","passwordHash","createdAt","updatedAt") VALUES ($1,$2,$3,$4) RETURNING *;

topgun@LAPTOP-I09H2UDN MINGW64 ~/Course3-Exercises/refactor-udagram (main)
```