Task 3 - Get it to work with Kubernetes

Environment:

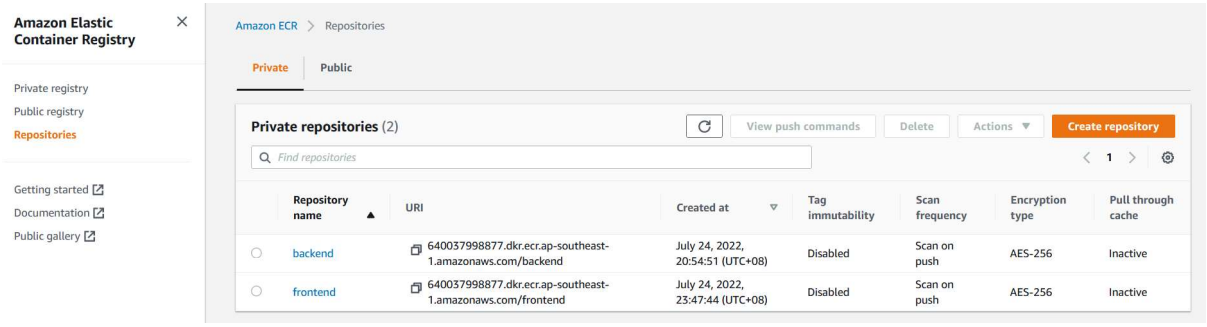
Ubuntu02 vm -> local docker repository running on virtualbox

AWS ECR -> remote repository

Minikube vm -> minikube running on virtualbox

1. Setup registry using AWS ECR

1a. Create repositories on AWS ECR



1b. Configure AWS credentials using programmatic user

```
necsa@ubuntu02:~$ aws configure list
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************OSAP shared-credentials-file
secret_key     ****************AUnj shared-credentials-file
    region           ap-southeast-1      config-file    ~/.aws/config
necsa@ubuntu02:~$
```

```
necsa@ubuntu02:~$ aws ecr get-login-password --region ap-southeast-1 | docker login --username AWS --password-stdin 640037
998877.dkr.ecr.ap-southeast-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/necsa/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
necsa@ubuntu02:~$
```

1c. Configure and enable minikube addons to connect to AWS ECR

Minikube addons configure registry-creds

Minikube addons enable registry-creds

```
Do you want to enable AWS Elastic Container Registry? [y/n]: y
-- Enter AWS Access Key ID: AKIAZKBJ42EOYKYPOSAP
-- Enter AWS Secret Access Key: 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
-- (Optional) Enter AWS Session Token:
-- Enter AWS Region: ap-southeast-1
-- Enter 12 digit AWS Account ID (Comma separated list): 640037998877
-- (Optional) Enter ARN of AWS role to assume:

Do you want to enable Google Container Registry? [y/n]: n

Do you want to enable Docker Registry? [y/n]: n

Do you want to enable Azure Container Registry? [y/n]: n
* registry-creds was successfully configured
PS C:\Users\shaun_king\Desktop\smartcow\task3> minikube addons enable registry-creds
  - Using image upmcenterprises/registry-creds:1.10
* The 'registry-creds' addon is enabled
PS C:\Users\shaun_king\Desktop\smartcow\task3>
```

1d. Tag local images to AWS ECR using below commands:

docker tag backend-flask:latest 640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/backend:latest

docker tag frontend-react:latest 640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/frontent:latest

```
necsa@ubuntu02: ~
necsa@ubuntu02:~$ docker images
REPOSITORY                                                  TAG       IMAGE ID       CREATED        SIZE
640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/frontend  latest    13bd3e804e68   3 hours ago    142MB
frontend-react                                              latest    13bd3e804e68   3 hours ago    142MB
proxy-nginx                                                 latest    f48f65fcaa29   24 hours ago   142MB
backend-flask                                               latest    51b240258b77   25 hours ago   930MB
640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/backend   latest    51b240258b77   25 hours ago   930MB
necsa@ubuntu02:~$
```

1e. Push tagged images to AWS ECR

```
necsa@ubuntu02: ~
necsa@ubuntu02:~$ docker push 640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/frontend
Using default tag: latest
The push refers to repository [640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/frontend]
e4b5472ee037: Layer already exists
c0100feae0ac: Layer already exists
5f70bf18a086: Layer already exists
e36a14968fa3: Layer already exists
35f05d5547f4: Layer already exists
42e6e8955651: Layer already exists
9f803fac20f7: Layer already exists
3a8f99f3bd90: Layer already exists
43b3c4e3001c: Layer already exists
latest: digest: sha256:cd1533bae0ba66c347fd9d31f86b5826886ebb05b8ad23a071359deee545ad70 size: 2193
necsa@ubuntu02:~$ docker push 640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/backend
Using default tag: latest
The push refers to repository [640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/backend]
a93b6184b45f: Layer already exists
8d426d0eda05: Layer already exists
6e5ee5d2595e: Layer already exists
9a1378e891b0: Layer already exists
e80c2603f92a: Layer already exists
37f7caf423fc: Layer already exists
1f8751be0506: Layer already exists
59b0c7a2fe4d: Layer already exists
7372faf8e603: Layer already exists
9be7f4e74e71: Layer already exists
36cd374265f4: Layer already exists
5bdeef4a08f3: Layer already exists
latest: digest: sha256:beb8c026adbdb467e2f03079aeb77f4f2ba4c1511a7b82105da1080d140e51be size: 2843
necsa@ubuntu02:~$
```

Amazon ECR > Repositories > backend

backend

View pu

Images (1)

Find images

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status
latest	Image	July 25, 2022, 13:37:28 (UTC+08)	357.06	Copy URI	sha256:beb8c026adbdb4...	Complete

Amazon ECR > Repositories > frontend

frontend

View pus

Images (1)

Find images

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status
latest	Image	July 25, 2022, 13:37:12 (UTC+08)	56.91	Copy URI	sha256:cd1533bae0ba66...	Complete

2. Enable minikube add-on ingress controller

(reference: https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/)

```
PS C:\Users\shaun_king> minikube addons enable ingress
  - Using image k8s.gcr.io/ingress-nginx/controller:v1.2.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

3. Backend Deployment

3a. Backend deployment yaml

backend-deploy.yaml 1

C: > Users > shaun_king > Desktop > smartcow > task3 > backend-deploy.yaml > {} spec

io.k8s.api.apps.v1.Deployment (v1@deployment.json)

```yaml

apiVersion: apps/v1
kind: Deployment

metadata:
  name: backend-deployment
spec:

  replicas: 1
  selector:
    matchLabels:
      component: backend

  template:
    metadata:
      labels:
        component: backend

    spec:
      containers:

        - name: backend
          image: 640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/backend:latest
          ports:
            - containerPort: 5000
```

3b. Apply backend-deploy.yaml

```
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl apply -f backend-deploy.yaml
deployment.apps/backend-deployment created
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl get deploy
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
backend-deployment    1/1     1            1           18s
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
backend-deployment-5895d9fb6c-d5ckp    1/1     Running   0          26s
C:\Users\shaun_king\Desktop\smartcow\task3>
```

3c. Describe backend pod

```
Admin: C:\Users\shaun_king\Desktop\smartcow\task3 - PowerShell 5.1 (9480)
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl describe pod backend-deployment-5895d9fb6c-d5ckp
Name:         backend-deployment-5895d9fb6c-d5ckp
Namespace:    default
Priority:     0
Node:         minikube/192.168.59.100
Start Time:   Tue, 26 Jul 2022 18:20:40 +0800
Labels:       component=backend
              pod-template-hash=5895d9fb6c
Annotations:  <none>
Status:       Running
IP:           172.17.0.5
IPs:
  IP:           172.17.0.5
Controlled By:  ReplicaSet/backend-deployment-5895d9fb6c
Containers:
  backend:
    Container ID:   docker://9e88e35b3a172431d7b40520d0110e731210af8dec077fe8ea2637dfeeb13aa4
    Image:          640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/backend:latest
    Image ID:       docker-pullable://640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/backend@sha256:beb8c026adbdb467e2f03079aeb77f4f2ba4c1511a7b82105da1080d140e51be
    Port:           5000/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 26 Jul 2022 18:20:42 +0800
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-sklbt (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-sklbt:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
```

4. Frontend Deployment

4a. Frontend deployment yaml

```yaml
! frontend-deploy.yaml 1 ×
C: > Users > shaun_king > Desktop > smartcow > task3 > ! frontend-deploy.yaml > {} spec > {} template
io.k8s.api.apps.v1.Deployment (v1@deployment.json)

apiVersion: apps/v1
kind: Deployment

metadata:
  name: frontend-deployment

spec:
  replicas: 1
  selector:
    matchLabels:
      component: frontend

  template:
    metadata:
      labels:
        component: frontend

    spec:
      containers:

      - name: frontend
        image: 640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/frontend:latest
        ports:
        - containerPort: 80
```

4b. Apply frontend-deploy.yaml

```
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl apply -f frontend-deploy.yaml
deployment.apps/frontend-deployment created
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl get deploy
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
backend-deployment    1/1     1            1           90s
frontend-deployment   1/1     1            1           6s
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
backend-deployment-5895d9fb6c-d5ckp    1/1     Running   0          94s
frontend-deployment-67b5667dfd-zsmsz   1/1     Running   0          9s
C:\Users\shaun_king\Desktop\smartcow\task3>
```

4c. Describe frontend pod

```
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl describe pod frontend-deployment-67b5667dfd-zsmsz
Name:         frontend-deployment-67b5667dfd-zsmsz
Namespace:    default
Priority:     0
Node:         minikube/192.168.59.100
Start Time:   Tue, 26 Jul 2022 18:22:05 +0800
Labels:       component=frontend
              pod-template-hash=67b5667dfd
Annotations:  <none>
Status:       Running
IP:           172.17.0.6
IPs:
  IP:           172.17.0.6
Controlled By:  ReplicaSet/frontend-deployment-67b5667dfd
Containers:
  frontend:
    Container ID:   docker://8c6d6097de409943ece0f95f777f9a4fe01bd50e19c98d588ada5654cb414c33
    Image:          640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/frontend:latest
    Image ID:       docker-pullable://640037998877.dkr.ecr.ap-southeast-1.amazonaws.com/frontend@sha256:12796f680ca6d691d679b15d66152116084fab6d1155bf1801a9e48a52f157c6
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 26 Jul 2022 18:22:06 +0800
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-rz2ts (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-rz2ts:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
```

5. Service

5a. Service yaml

```yaml
! service.yaml ×
C: > Users > shaun_king > Desktop > smartcow > task3 > ! service.yaml > apiVersion
io.k8s.api.core.v1.Service (v1@service.json)
apiVersion: v1

kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    component: frontend
  type: ClusterIP
  ports:
    - port: 80
      targetPort: 80

---
apiVersion: v1

kind: Service
metadata:
  name: backend-service
spec:
  selector:
    component: backend
  type: ClusterIP
  ports:
    - port: 80
      targetPort: 5000
```

5c. Apply service.yaml

```
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl apply -f service.yaml
service/frontend-service created
service/backend-service created
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl get svc
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
backend-service     ClusterIP   10.105.149.181   <none>        80/TCP    10s
frontend-service    ClusterIP   10.103.100.136   <none>        80/TCP    10s
kubernetes          ClusterIP   10.96.0.1        <none>        443/TCP   33h
C:\Users\shaun_king\Desktop\smartcow\task3>
```

6. Ingress Controller:

6a. Ingress controller yaml

```
io.k8s.api.networking.v1.Ingress (v1@ingress.json)
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-controller
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/use-regex: "true"
    nginx.ingress.kubernetes.io/rewrite-target: /$1

spec:
  rules:
    - http:
        paths:
          - path: /?(.*)
            pathType: Prefix
            backend:
              service:
                name: frontend-service
                port:
                  number: 80
```

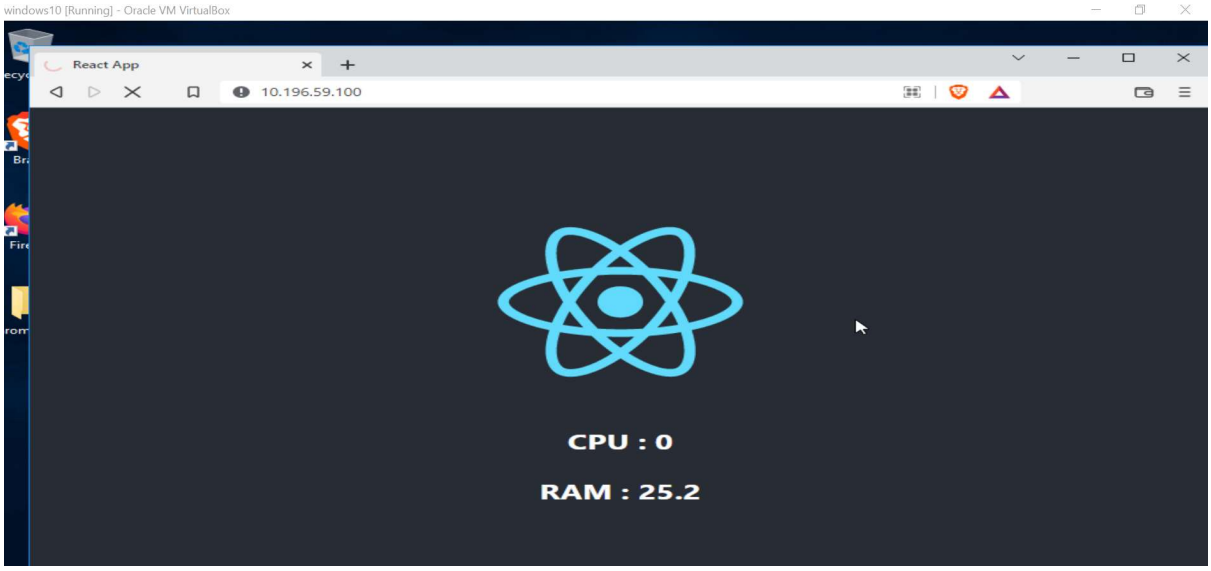
6b. Apply ingress.yaml

```
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl apply -f ingress.yaml
ingress.networking.k8s.io/ingress-controller created
```

6c. Describe ingress

```
C:\Users\shaun_king\Desktop\smartcow\task3> kubectl describe ing ingress-controller
Name:             ingress-controller
Labels:           <none>
Namespace:        default
Address:          192.168.59.100
Ingress Class:    <none>
Default backend:  <default>
Rules:
  Host        Path  Backends
  ----        ----  --------
  *
              /?(.*)   frontend-service:80 (172.17.0.6:80)
Annotations:  kubernetes.io/ingress.class: nginx
              nginx.ingress.kubernetes.io/rewrite-target: /$1
              nginx.ingress.kubernetes.io/use-regex: true
Events:
  Type    Reason  Age                From                      Message
  ----    ------  ----               ----                      -------
  Normal  Sync    37s (x2 over 86s)  nginx-ingress-controller  Scheduled for sync
C:\Users\shaun_king\Desktop\smartcow\task3>
```

7. **<u>Test application using Minikube exposed IP</u>**


windows10 [Running] - Oracle VM VirtualBox
React App
10.196.59.100
CPU : 0
RAM : 25.2