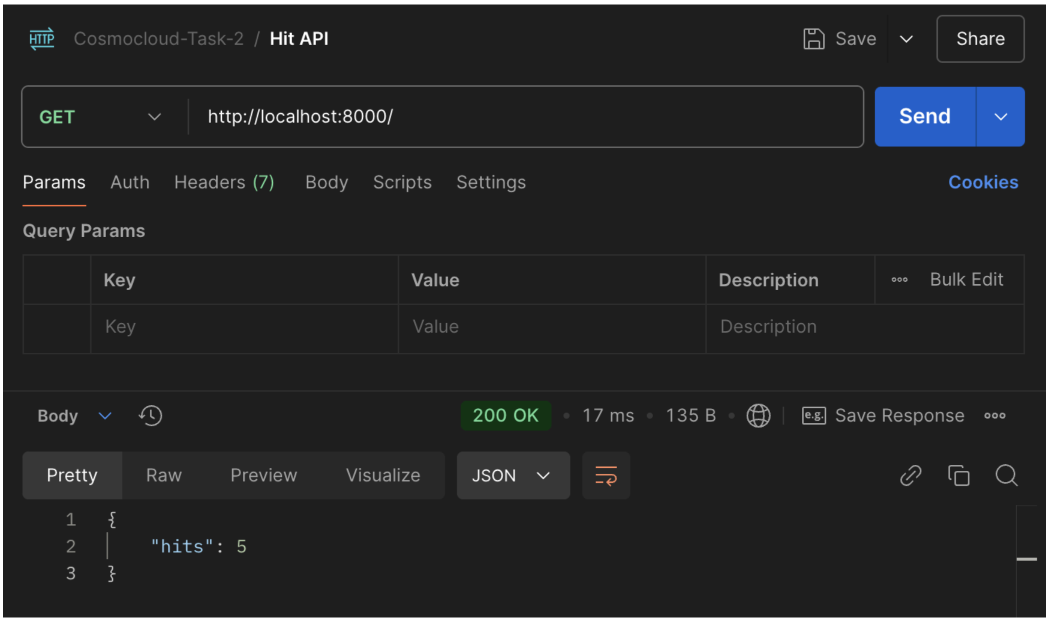
Cosmocloud Task 2

Debug & Fix Application/Helm Chart

By- Jay Oswal

Working API Call Screenshots

1. Postman



Cosmocloud-Task-2 / Hit API

GET http://localhost:8000/

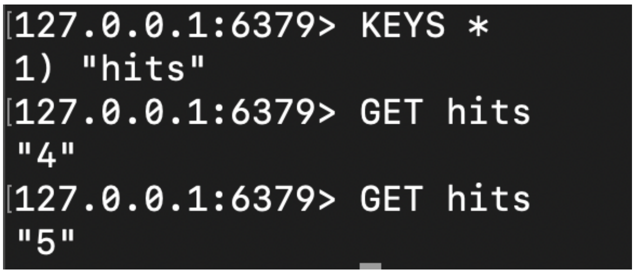
Query Params

Key	Value	Description
Key	Value	Description

200 OK • 17 ms • 135 B

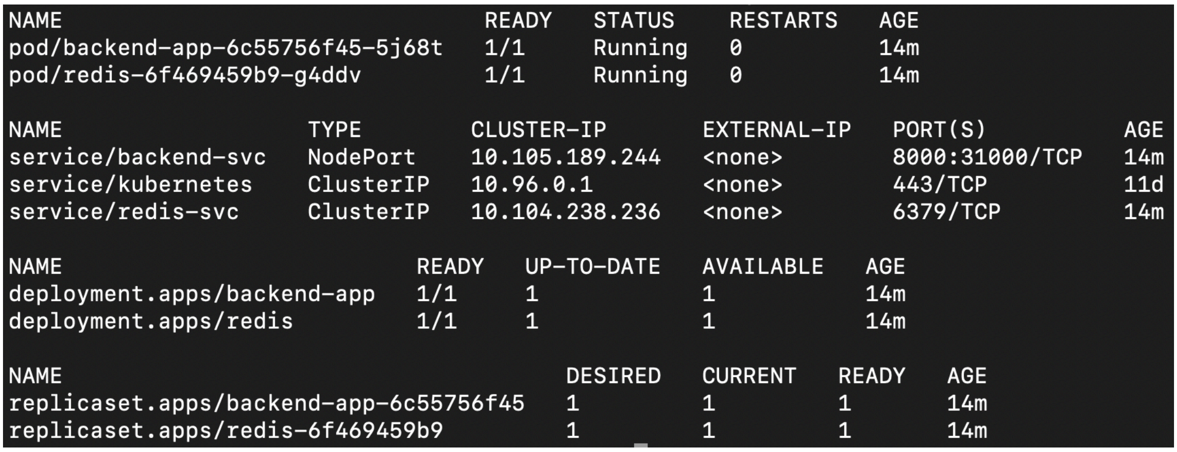
```
{
    "hits": 5
}
```

2. Redis through CLI
   a. kubectl exec -it <redis-pod-name> -- redis-cli



```
127.0.0.1:6379> KEYS *
1) "hits"
127.0.0.1:6379> GET hits
"4"
127.0.0.1:6379> GET hits
"5"
```

3. ‘*kubectl get all*’



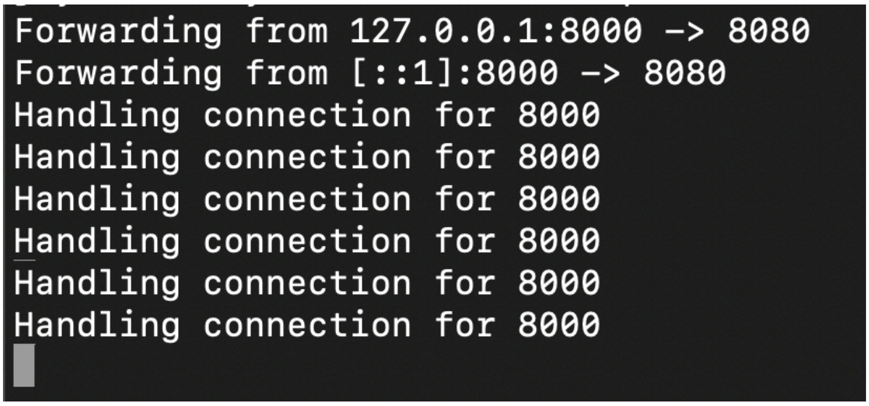
```
NAME                                READY   STATUS    RESTARTS   AGE
pod/backend-app-6c55756f45-5j68t   1/1     Running   0          14m
pod/redis-6f469459b9-g4ddv         1/1     Running   0          14m

NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/backend-svc   NodePort    10.105.189.244   <none>        8000:31000/TCP   14m
service/kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP          11d
service/redis-svc     ClusterIP   10.104.238.236   <none>        6379/TCP         14m

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/backend-app   1/1     1            1           14m
deployment.apps/redis         1/1     1            1           14m

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/backend-app-6c55756f45   1         1         1       14m
replicaset.apps/redis-6f469459b9         1         1         1       14m
```

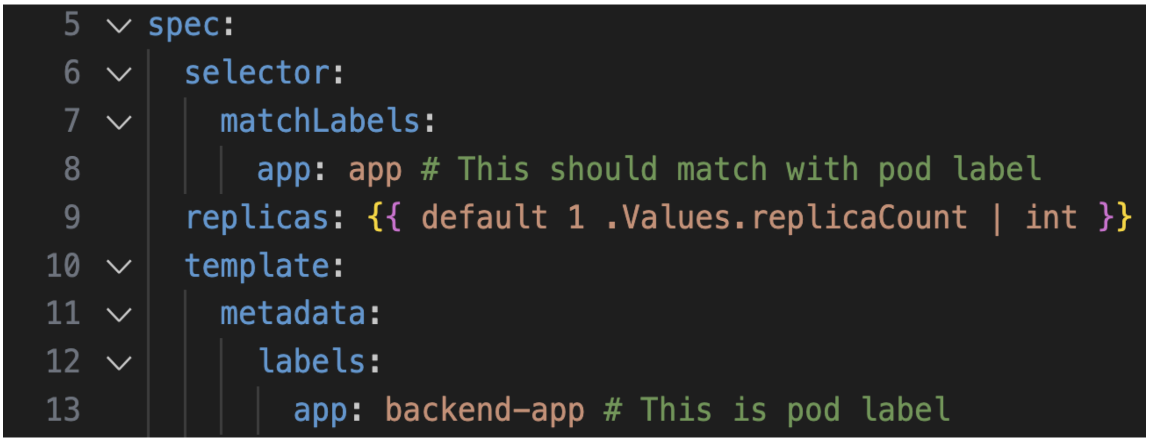
4. Kubernetes port forwarding → ‘*kubectl port-forward svc/backend-svc 8000:8000*’



```
Forwarding from 127.0.0.1:8000 -> 8080
Forwarding from [::1]:8000 -> 8080
Handling connection for 8000
Handling connection for 8000
Handling connection for 8000
Handling connection for 8000
Handling connection for 8000
Handling connection for 8000
```

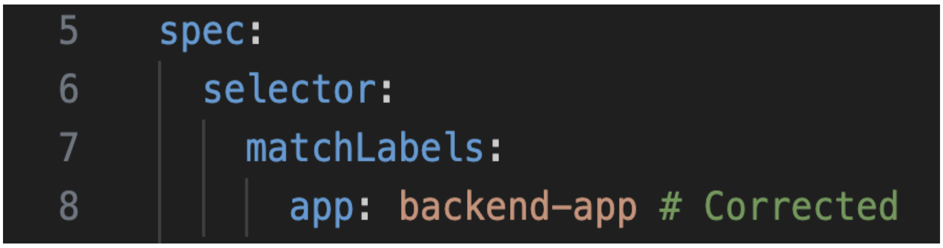
Total Bugs Found: 10

1. Bug 1
   a. File: deployment.yaml
   b. How did I find this?: By Executing command ‘helm install sample-helm-chart .’
   c. Error: *Deployment.apps "backend-app" is invalid: spec.template.metadata.labels: Invalid value: map[string]string{"app":"backend-app"}: `selector` does not match template `labels`*



```
spec:
  selector:
    matchLabels:
      app: app # This should match with pod label
  replicas: {{ default 1 .Values.replicaCount | int }}
  template:
    metadata:
      labels:
        app: backend-app # This is pod label
```

   d. Explanation: The deployment uses matchLabels to manage the correct set of pods. If it doesn’t matches the deployment will be unable to manage pods and service will not route traffic correctly.
   e. Solution : Use the same pod name in the selector



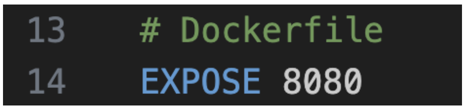
```
spec:
  selector:
    matchLabels:
      app: backend-app # Corrected
```

2. Bug 2

a. File: deployment.yaml

b. How did I find this?: By Observation

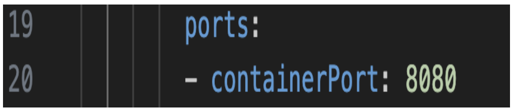
c. Error: As per Docker file of the backend application port ‘8080’ is supposed to be exposed. So in deployment.yaml file the container port is mentioned incorrectly as 8000



```
# Dockerfile
EXPOSE 8080
```

d. Explanation: There will be port conflic leading to networking & request routing issues as the container’s port do not match(8000) to what it is exposed(8080).

e. Solution : Keep port consistent to avoid routing traffic & networking issues. So using Dockerfile as source of truth, all related port is used as 8080



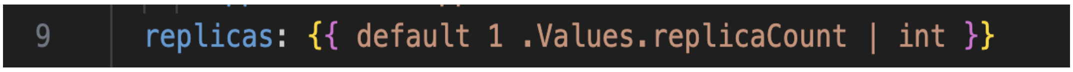
```
      ports:
      - containerPort: 8080
```

3. Bug 3

a. File: deployment.yaml

b. How did I find this?: By Observation

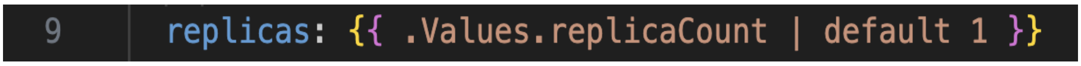
c. Error: Incorrect syntax for default when values not specified in values.yaml



```
  replicas: {{ default 1 .Values.replicaCount | int }}
```

d. Explanation: Possible incorrect syntax

e. Solution: Changed to recommended syntax to handle default



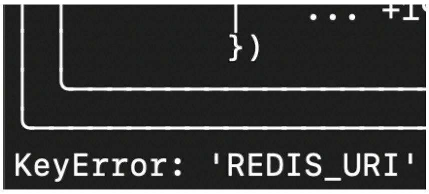
```
  replicas: {{ .Values.replicaCount | default 1 }}
```

4. Bug 4

a. File: deployment.yaml

b. How did I find this?: By checking log of backend pod, when pod failed to start and went into CrashLoopBackOff. By using command ‘*kubectl logs <backend pod name>*’

c. Error: REDIS_URI not specified in env

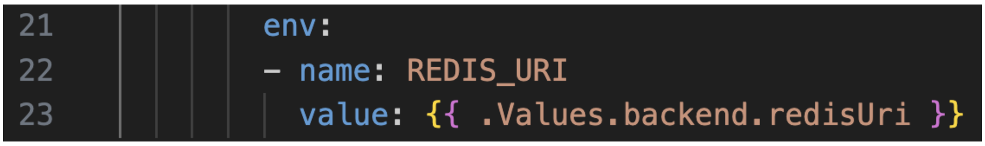


```
                ... +1
            })
KeyError: 'REDIS_URI'
```

d. Explanation: The backend image needs REDIS_URI as it looks into env, failing to specified will not start the backend pod

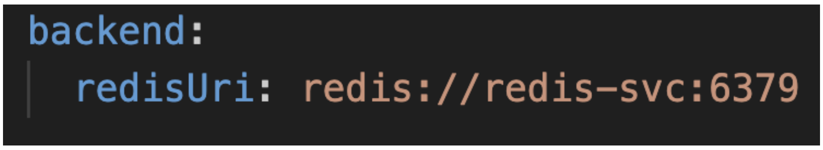
e. Solution: Mention it in the deployment file through the values.yaml file

i. deployment.yaml



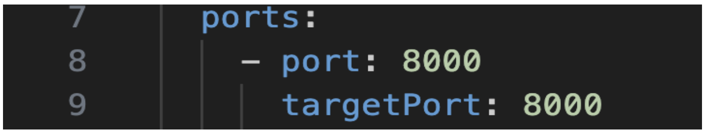
```
        env:
        - name: REDIS_URI
          value: {{ .Values.backend.redisUri }}
```

ii. values.yaml



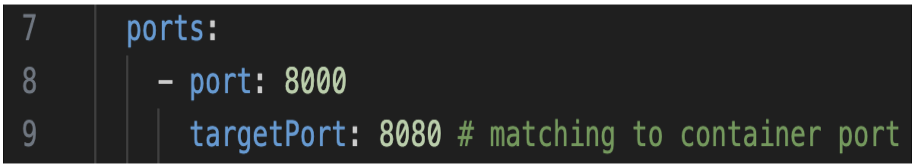
```
backend:
  redisUri: redis://redis-svc:6379
```

5. Bug 5
    a. File: backend-svc.yaml
    b. How did I find this?: By Observation
    c. Error: Incorrect port for service. Target port not matching to exposed port of backend image (Dockerfile)



```yaml
ports:
  - port: 8000
    targetPort: 8000
```

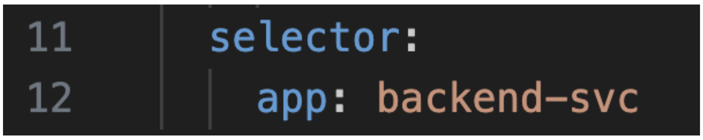
    d. Explanation: This will lead to incorrect routing of traffic to non-desired port.
    e. Solution: Match the port to that of Image's port



```yaml
ports:
  - port: 8000
    targetPort: 8080 # matching to container port
```

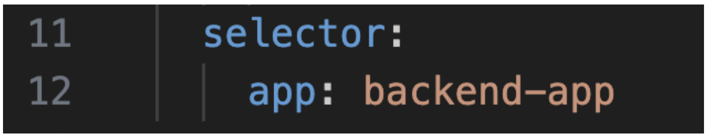
Note:- As on which port should the backend app be accessible from outside has not been mentioned, I am using port 8000 as service port

6. Bug 6
    a. File: backend-svc.yaml
    b. How did I find this?: By Observation
    c. Error: Incorrect selector for backend app. We've named the pod as 'backend-app' and not 'backend-svc'



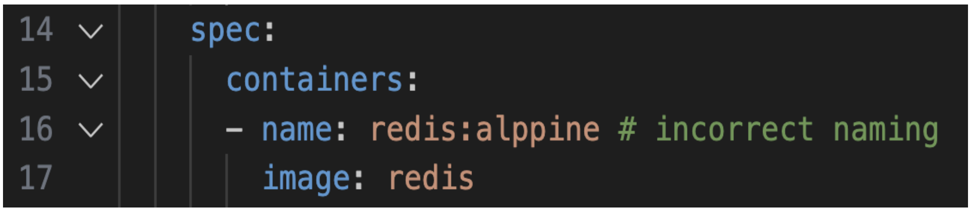
```yaml
selector:
  app: backend-svc
```

    d. Explanation: Selector should match with our correct desired backend app name, else will result in incorrect network routing.
    e. Solution: Match the app selector to correct pod name



```yaml
selector:
  app: backend-app
```

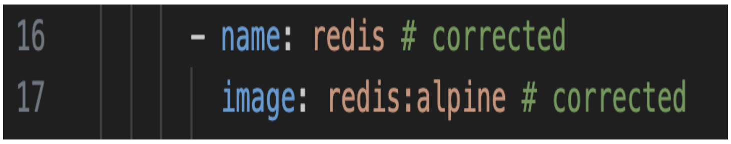
7. Bug 7
    a. File: redis.yaml
    b. How did I find this?: By Executing command '*helm install sample-helm-chart .*'
    c. Error: *Deployment.apps "redis" is invalid: spec.template.spec.containers[0].name: Invalid value: "redis:alpine": a lowercase RFC 1123 label must consist of lower case alphanumeric characters or '-', and must start and end with an alphanumeric character (e.g. 'my-name', or '123-abc', regex used for validation is '[a-z0-9]([-a-z0-9]*[a-z0-9])?')*



```yaml
spec:
  containers:
  - name: redis:alppine # incorrect naming
    image: redis
```

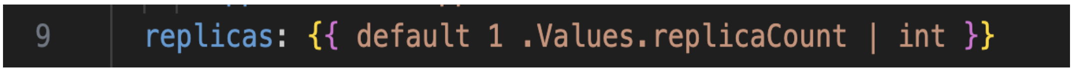
d. Explanation: Incorrect naming convention for container OR container’s name & image name are swapped.

e. Solution : Just kept name as redis and image name as redis:alpine



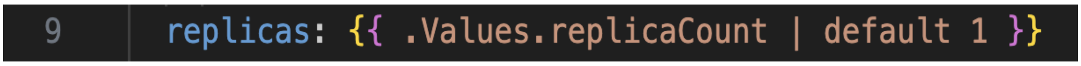
```
      - name: redis # corrected
        image: redis:alpine # corrected
```

8. Bug 8
   a. File: redis.yaml
   b. How did I find this?: By Observation
   c. Error: Incorrect syntax for default when values not specified in values.yaml



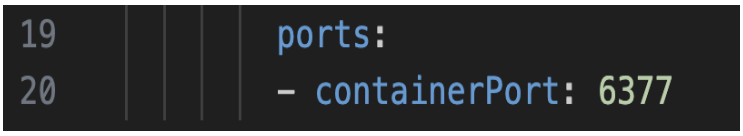
```
  replicas: {{ default 1 .Values.replicaCount | int }}
```

   d. Explanation: Possible incorrect syntax
   e. Solution: Changed to recommended syntax to handle default



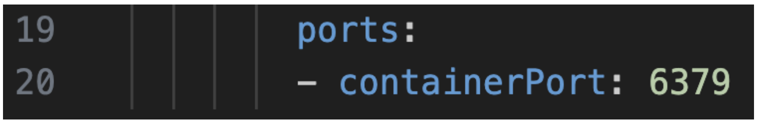
```
  replicas: {{ .Values.replicaCount | default 1 }}
```

9. Bug 9
   a. File: redis.yaml
   b. How did I find this?: By Observation
   c. Error: Incorrect container port



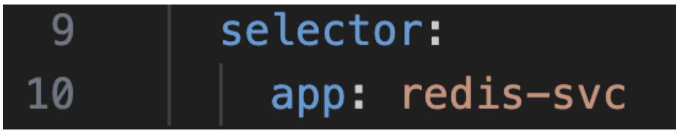
```
        ports:
        - containerPort: 6377
```

   d. Explanation: We are using default port of redis which is 6379 in the redis docker image. So 6377 will not point to redis image resulting in networking issues
   e. Solution: Use correct redis port in its deployment file



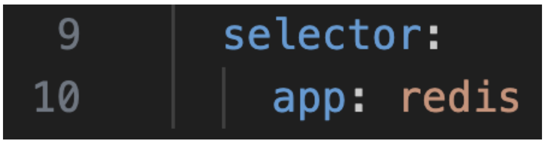
```
        ports:
        - containerPort: 6379
```

10.Bug 10
   a. File: redis-svc.yaml
   b. How did I find this?: By Observation
   c. Error: Incorrect app selector name



```
  selector:
    app: redis-svc
```

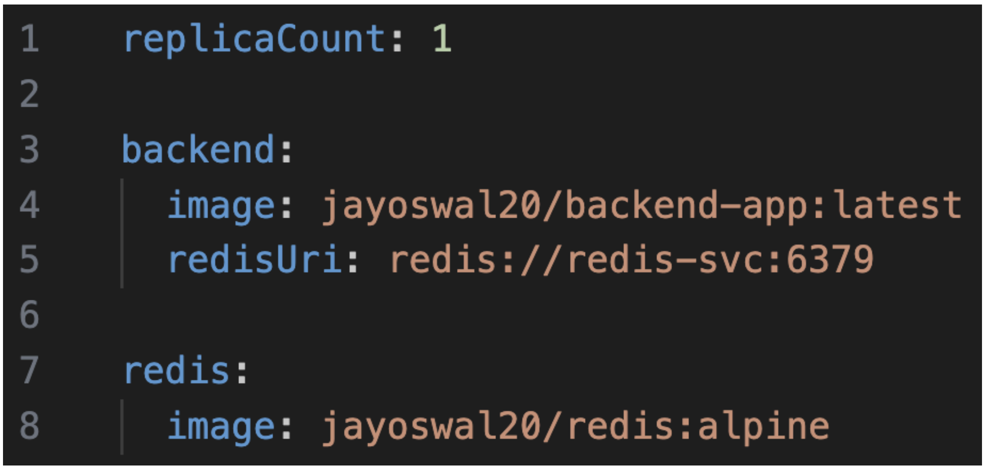
   d. Explanation: We are using redis as our pod name for redis container hence, using redis-svc will result in incorrect network routing.
   e. Solution: Use correct redis pod name



```
  selector:
    app: redis
```

Values.yaml File

I utilized values.yaml file for image name and redis url as it will easy to maintain for CI/CD purpose. I also pushed the images into my docker repository ‘jayoswal20’ as it was already configured on my machine, else I had to point minikube to local docker repository.

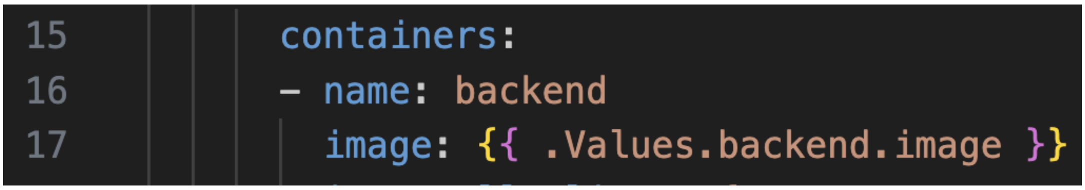


```yaml
replicaCount: 1

backend:
  image: jayoswal20/backend-app:latest
  redisUri: redis://redis-svc:6379

redis:
  image: jayoswal20/redis:alpine
```

Pointed to values.yaml wherever applicable



```yaml
      containers:
      - name: backend
        image: {{ .Values.backend.image }}
```

Github Repo with bug fixed Helm Charts → https://github.com/jayoswal/DevOps-Task-2