Al-Ahly Momkn
Fintech
DevOps department

Summer internship
August 2025
Omar M. Gabr

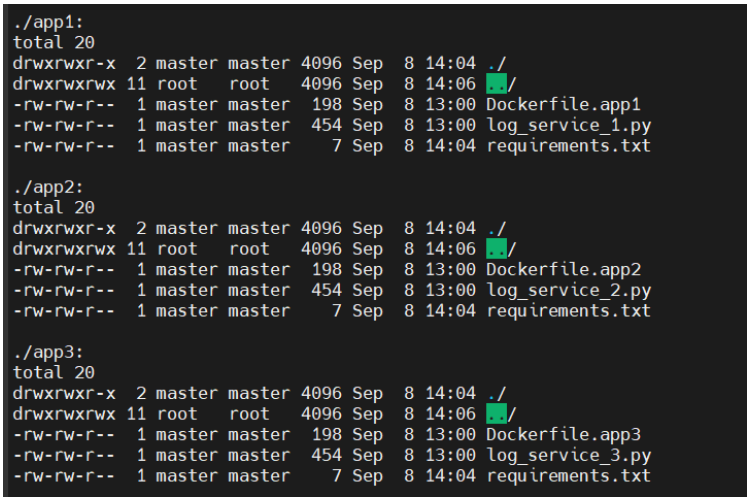
Task2

Helm, Persistent Storage and Ingress

Step1: Containerization

Containerization

Each service moved to a folder



Build Commands

Then containerized using following docker commands for each:

```
$ docker build -f Dockerfile.app1 -t log-service-1 .
$ docker build -f Dockerfile.app2 -t log-service-2 .
$ docker build -f Dockerfile.app3 -t log-service-3 .
```

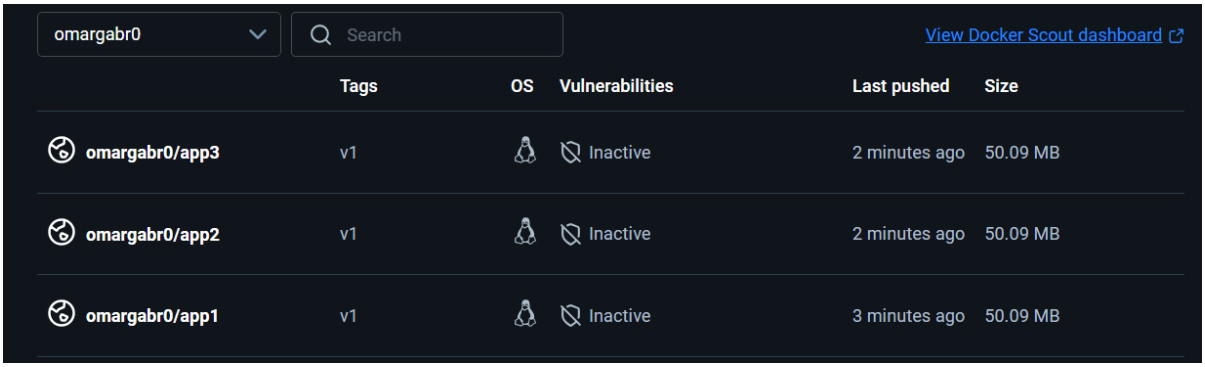
Tagging

```
$ docker tag log-service-1 omargabr0/app1:v1
$ docker tag log-service-2 omargabr0/app2:v1
$ docker tag log-service-3 omargabr0/app3:v1
```

Pushing to docker hub

```
$ docker push omargabr0/app1:v1
$ docker push omargabr0/app2:v1
$ docker push omargabr0/app3:v1
```

Docker hub snapshot



Step2: Helm Deployment

App1 Folder structure

```
task2/
└── app1/
    ├── Chart.yaml
    ├── values.yaml
    ├── templates/
    │   ├── deployment.yaml
    │   ├── service.yaml
    │   ├── ingress.yaml
    │   ├── hpa.yaml
    │   ├── _helpers.tpl
    │   ├── NOTES.txt
    │   └── tests/
    │       └── test-connection.yaml
    └── charts/
```

App1:

Values.yaml:

```yaml
replicaCount: 1

image:
  repository: omargabr0/app1
  tag: v1
  pullPolicy: IfNotPresent

namespace: env-1

service:
  type: NodePort
```

```
  port: 5001
  nodePort: 30011

hpa:
  enabled: true
  minReplicas: 1
  maxReplicas: 5
  CPU:
    averageUtilization: 70
  memory:
    enabled: false
    targetMemoryUtilizationPercentage: 80

ingress:
  enabled: false

serviceAccount:
  create: false
```

Deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}
  labels:
    app: {{ .Release.Name }}
  namespace: {{ .Values.namespace }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: {{ .Release.Name }}
  template:
    metadata:
      labels:
        app: {{ .Release.Name }}
    spec:
      containers:
        - name: {{ .Release.Name }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          ports:
            - containerPort: {{ .Values.service.port }}
          resources:
            requests:
              cpu: "100m"
            limits:
```

```
            cpu: "500m"
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}
  labels:
    app: {{ .Release.Name }}
spec:
  type: {{ .Values.service.type }}
  selector:
    app: {{ .Release.Name }}
  ports:
    - protocol: TCP
      port: {{ .Values.service.port }}
      targetPort: {{ .Values.service.port }}
      {{- if eq .Values.service.type "NodePort" }}
      nodePort: {{ .Values.service.nodePort }}
      {{- end }}
```

hpa.yaml

```
{{- if .Values.hpa.enabled }}
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: {{ .Release.Name }}
  labels:
    app: {{ .Release.Name }}
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: {{ .Release.Name }}
  minReplicas: {{ .Values.hpa.minReplicas }}
  maxReplicas: {{ .Values.hpa.maxReplicas }}
  metrics:
    - type: Resource
      resource:
        name: cpu
        target:
          type: Utilization
          averageUtilization: {{ .Values.hpa.CPU.averageUtilization}}
```

```
    {{- if .Values.hpa.memory.enabled }}
    - type: Resource
      resource:
        name: memory
        target:
          type: Utilization
          averageUtilization: {{ .Values.hpa.targetMemoryUtilizationPercentage }}
    {{- end }}

{{- end }}
```

command outputs showing successful deployment.

```
master@master:/usr/new/task2/app1$ helm install app1-release1 ./app1
NAME: app1-release1
LAST DEPLOYED: Tue Sep  9 11:00:20 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services app1-release1)
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
master@master:/usr/new/task2/app1$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
app1-release1   default         1               2025-09-09 11:00:20.459538217 +0000 UTC deployed        app1-0.1.0      1.0
master@master:/usr/new/task2/app1$ helm history app1-release1
REVISION        UPDATED                         STATUS          CHART           APP VERSION     DESCRIPTION
1               Tue Sep  9 11:00:20 2025        deployed        app1-0.1.0      1.0             Install complete
master@master:/usr/new/task2/app1$
```

App2 Folder structure

```
task2/
└── app2/
    ├── Chart.yaml
    ├── values-env1.yaml
    ├── values-env2.yaml
    ├── templates/
    │   ├── deployment.yaml
    │   ├── service.yaml
    │   ├── _helpers.tpl
    │   ├── NOTES.txt
    │   └── tests/
    │       └── test-connection.yaml
    └── charts/
```

Values-env1.yaml

```
replicaCount: 1

image:
  repository: omargabr0/app2
  tag: v1
```

```
  pullPolicy: IfNotPresent

namespace: env-1

service:
  type: NodePort
  port: 5002
  nodePort: 30012

hpa:
  enabled: true
  minReplicas: 1
  maxReplicas: 5
  CPU:
    averageUtilization: 70
  memory:
    enabled: false
    targetMemoryUtilizationPercentage: 80

ingress:
  enabled: false

serviceAccount:
  create: false
```

Values-env2.yaml

```
replicaCount: 1

image:
  repository: omargabr0/app2
  tag: v1
  pullPolicy: IfNotPresent

namespace: env-2

service:
  type: NodePort
  port: 5002
  nodePort: 30013

hpa:
  enabled: true
  minReplicas: 1
  maxReplicas: 5
  CPU:
    averageUtilization: 70
  memory:
    enabled: false
```

```
    targetMemoryUtilizationPercentage: 80

ingress:
  enabled: false

serviceAccount:
  create: false
```

rest of templet files as deployment.yaml, service.yaml, hpa.yaml are the same as app1, only changes where necessary to be made are in the values.yaml.

Command to run using specified values files:

$ helm install app2-env1 ./app2 -f /usr/new/task2/app2/app2/values-env1.yaml

$ helm install app2-env2 ./app2 -f /usr/new/task2/app2/app2/values-env2.yaml

command outputs showing successful deployment.

```
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services app2-env1)
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
master@master:/usr/new/task2/app2$ kubectl get pods -n env-2
NAME                         READY   STATUS    RESTARTS   AGE
app2-env2-bfcb4fbcb-vm4g4    1/1     Running   0          67s
master@master:/usr/new/task2/app2$
```

```
master@master:/usr/new/task2/app2$ helm status app2-env1
NAME: app2-env1
LAST DEPLOYED: Tue Sep  9 12:38:55 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services app2-env1)
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
master@master:/usr/new/task2/app2$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
app1-release1   default         1               2025-09-09 11:00:20.459538217 +0000 UTC deployed        app1-0.1.0      1.0
app2-env1       default         1               2025-09-09 12:38:55.872008943 +0000 UTC deployed        app2-0.1.0      1.16.0
master@master:/usr/new/task2/app2$ kubectl get pods -n env2
```

App3: Folder structure

Similar to app1

```
task2/
└── app3/
    ├── Chart.yaml
    ├── values.yaml
    ├── templates/
    │   ├── deployment.yaml
    │   ├── service.yaml
```

```
    │   ├── ingress.yaml
    │   ├── hpa.yaml
    │   ├── _helpers.tpl
    │   ├── NOTES.txt
    │   └── tests/
    │       └── test-connection.yaml
    └── charts/
```

Values.yaml

```yaml
replicaCount: 1

image:
  repository: omargabr0/app3
  tag: v1
  pullPolicy: IfNotPresent

namespace: env-3

service:
  type: NodePort
  port: 5003
  nodePort: 30013

hpa:
  enabled: true
  minReplicas: 1
  maxReplicas: 5
  CPU:
    averageUtilization: 70
  memory:
    enabled: false
    targetMemoryUtilizationPercentage: 80

ingress:
  enabled: false

serviceAccount:
  create: false
```

command outputs showing successful deployment.

```
release "app3-env3" uninstalled
master@master:/usr/new/task2/app3$ helm install app3-env2 ./app3
NAME: app3-env2
LAST DEPLOYED: Tue Sep  9 14:16:54 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services app3-env2)
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
master@master:/usr/new/task2/app3$ kubectl get pods -n env2
No resources found in env2 namespace.
master@master:/usr/new/task2/app3$ kubectl get pods -n env-2
NAME                         READY   STATUS    RESTARTS   AGE
app2-env2-bfcb4fbcb-vm4g4    1/1     Running   0          91m
app3-env2-54ddf68b44-5r7r6   1/1     Running   0          17s
master@master:/usr/new/task2/app3$ kubectl get svc -n env-2
No resources found in env-2 namespace.
master@master:/usr/new/task2/app3$ kubectl get svc
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
app1-release1   NodePort    10.110.190.138   <none>        5001:30011/TCP   3h17m
app2-env1       NodePort    10.111.220.101   <none>        5002:30012/TCP   98m
app2-env2       NodePort    10.103.237.36    <none>        5002:30013/TCP   91m
app3-env2       NodePort    10.100.244.87    <none>        5003:30014/TCP   39s
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP          5d15h
ner-svc         NodePort    10.96.186.221    <none>        80:30002/TCP     4d1h
proxy-svc       NodePort    10.106.12.174    <none>        80:30003/TCP     4d1h
sentiment-svc   NodePort    10.111.180.255   <none>        80:30001/TCP     4d1h
master@master:/usr/new/task2/app3$
```

Step 3: Persistent Storage

Creating pv-pvc file in app/template/pv-pvc.yaml so that it will be implemented to helm chart.

App1

Adding pv-pvc:

This will mount the volume to /logs in the container filesystem

And the the host store the pv data `/usr/new/task2/app1/logs`

Values.yaml

```yaml
pv:
  enabled: true
  accessMode: ReadWriteMany
  hostPath: /usr/new/task2/app3/logs
  mountPath: /logs
  persistentVolumeReclaimPolicy: Delete
  storage: 500Mi
```

Deployment.yaml

```yaml
          {{- if .Values.pv.enabled }}
          volumeMounts:
            - name: log-volume
              mountPath: {{ .Values.pv.mountPath }}
          {{- end }}
      {{- if .Values.pv.enabled }}
      volumes:
```

```
        - name: log-volume
          persistentVolumeClaim:
            claimName: {{ .Release.Name }}-pvc
      {{- end }}
```

pv-pvc.yaml

```
{{- if .Values.pv.enabled }}
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: {{ .Release.Name }}-pv
spec:
  capacity:
    storage: {{ .Values.pv.storage }}
  accessModes:
    - {{ .Values.pv.accessMode }}
  hostPath:
    path: {{ .Values.pv.hostPath  }}

---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: {{ .Release.Name }}-pvc
spec:
  accessModes:
    - {{ .Values.pv.accessMode }}
  resources:
    requests:
      storage: {{ .Values.pv.storage }}
{{- end }}
```

App2 in env-1 and env-2

Same yaml files, but the difference in values file.

Values-env1.yaml

```

pv:
  enabled: true
  accessMode: ReadWriteMany
  hostPath: /usr/new/task2/app2/logs-env1
  mountPath: /logs
```

```
  persistentVolumeReclaimPolicy: Delete
  storage: 500Mi
```

Values-env1.yaml

```
pv:
  enabled: true
  accessMode: ReadWriteMany
  hostPath: /usr/new/task2/app2/logs-env2
  mountPath: /logs
  persistentVolumeReclaimPolicy: Delete
  storage: 500Mi
```

App3

Creating pv-pvc file in app/template/pv-pvc.yaml so that it will be implemented to helm chart.

Values.yaml

```
pv:
  enabled: true
  accessMode: ReadWriteMany
  hostPath: /usr/new/task2/app3/logs
  mountPath: /logs
  persistentVolumeReclaimPolicy: Delete
  storage: 500Mi
```

Final helm application and pv- pvc snapshots:

```
master@master:~$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
app1-env1       default         1               2025-09-10 12:36:37.437597322 +0000 UTC deployed        app1-0.1.0      1.0
app2-env1       default         1               2025-09-10 12:24:36.055245865 +0000 UTC deployed        app3-0.1.0      1.0
app2-env2       default         1               2025-09-10 12:24:42.123441351 +0000 UTC deployed        app3-0.1.0      1.0
app3-env3       default         1               2025-09-10 12:33:09.911245765 +0000 UTC deployed        app3-0.1.0      1.0
master@master:~$ kubectl get pv && kubectl get pvc
NAME           CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
app1-env1-pv   500Mi      RWX            Delete           Bound    default/app1-env1-pvc                  <unset>                          5h1m
app2-env1-pv   500Mi      RWX            Delete           Bound    default/app2-env1-pvc                  <unset>                          5h13m
app2-env2-pv   500Mi      RWX            Delete           Bound    default/app2-env2-pvc                  <unset>                          5h13m
app3-env3-pv   500Mi      RWX            Delete           Bound    default/app3-env3-pvc                  <unset>                          5h5m
NAME            STATUS   VOLUME         CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
app1-env1-pvc   Bound    app1-env1-pv   500Mi      RWX                           <unset>                 5h1m
app2-env1-pvc   Bound    app2-env1-pv   500Mi      RWX                           <unset>                 5h13m
app2-env2-pvc   Bound    app2-env2-pv   500Mi      RWX                           <unset>                 5h13m
app3-env3-pvc   Bound    app3-env3-pv   500Mi      RWX                           <unset>                 5h5m
master@master:~$ kubectl get svc -n env-1 env-2
Error from server (NotFound): services "env-2" not found
master@master:~$ kubectl get svc -n env-1 -n env-2 -n env-3
NAME        TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
app3-env3   NodePort   10.100.1.90    <none>        5003:30014/TCP   5h5m
master@master:~$ kubectl get svc -n env-1
NAME        TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
app1-env1   NodePort   10.106.40.162    <none>        5001:30011/TCP   5h2m
app2-env1   NodePort   10.111.250.191   <none>        5002:30012/TCP   5h14m
master@master:~$ kubectl get svc -n env-2
NAME        TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
app2-env2   NodePort   10.111.59.123   <none>        5002:30013/TCP   5h14m
master@master:~$ kubectl get svc -n env-3
NAME        TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
app3-env3   NodePort   10.100.1.90    <none>        5003:30014/TCP   5h6m
```

This snapshot showes:

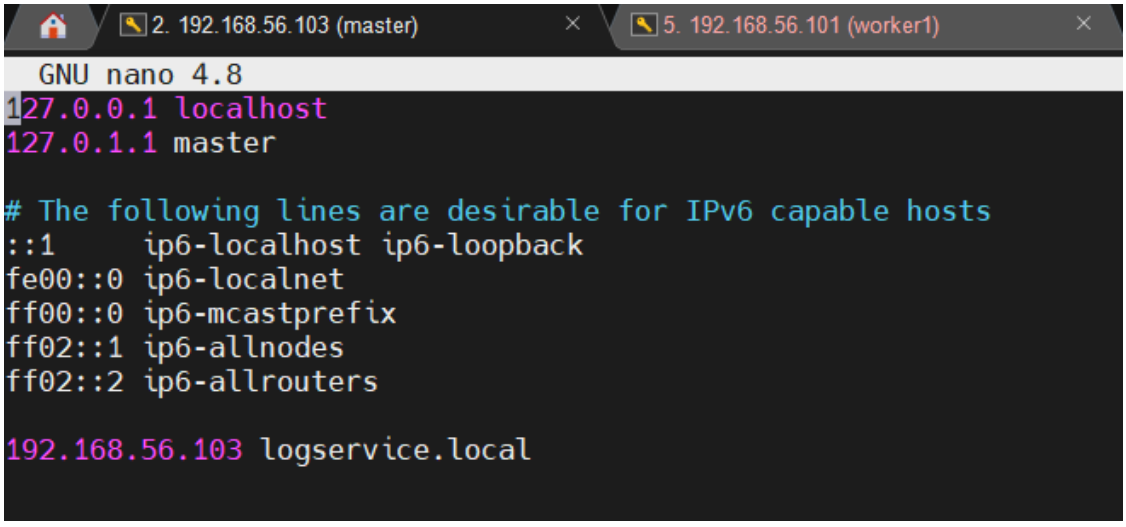
- all helm application
- pv and pvc created successfully
- services for each application

Step 4: Ingress

DNS configuration

This maps the master IP to domain so that we can use in ingress resource file

$ sudo nano /etc/hosts



```
master@master:/usr/new/task2/ingress$ kubectl apply -f ingress.yaml
ingress.networking.k8s.io/my-apps-ingress created
master@master:/usr/new/task2/ingress$ kubectl get validatingwebhookconfiguration ingress-nginx-admission -o yaml | grep failurePolicy
      {"apiVersion":"admissionregistration.k8s.io/v1","kind":"ValidatingWebhookConfiguration","metadata":{"annotations":{},"labels":{"app.kubernetes.io/compo
nent":"admission-webhook","app.kubernetes.io/instance":"ingress-nginx","app.kubernetes.io/name":"ingress-nginx","app.kubernetes.io/part-of":"ingress-nginx","
app.kubernetes.io/version":"1.13.2"},"name":"ingress-nginx-admission"},"webhooks":[{"admissionReviewVersions":["v1"],"clientConfig":{"service":{"name":"ingre
ss-nginx-controller-admission","namespace":"ingress-nginx","path":"/networking/v1/ingresses","port":443}},"failurePolicy":"Fail","matchPolicy":"Equivalent","
name":"validate.nginx.ingress.kubernetes.io","rules":[{"apiGroups":["networking.k8s.io"],"apiVersions":["v1"],"operations":["CREATE","UPDATE"],"resources":["
ingresses"]}],"sideEffects":"None"}]}
  failurePolicy: Ignore
```

Applying ingress

```
master@master:/usr/new/task2/ingress$ kubectl get ingress -A
NAMESPACE   NAME              CLASS   HOSTS             ADDRESS   PORTS   AGE
default     my-apps-ingress   nginx   logservice.loc              80      17m
master@master:/usr/new/task2/ingress$
```

1. Fix service

NodePort → Cluster IP

Exposing services using NodePort will break the routing as ingress forwards traffic inside the cluster directly to service endpoints.

2. Using ExternalName service

- create aliases of services in each namespace inside the same namespace as the Ingress so that ingress can see it from its namespace

- aliases points to the services in other namespaces like env-1, and ingress routes to the alaises

```yaml
apiVersion: v1
kind: Service
metadata:
  name: app1-env1-alias
  namespace: default
spec:
  type: ExternalName
  externalName: app1-env1.env-1.svc.cluster.local
  ports:
    - port: 5001
      protocol: TCP
```

3. ingress.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-apps-ingress
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/rewrite-target: /log
spec:
  rules:
  - host: logservice.local
    http:
      paths:
      - path: /app1
        pathType: Prefix
        backend:
          service:
            name: app1-env1-alias
            port:
              number: 5001
      - path: /app2/env1
        pathType: Prefix
        backend:
          service:
            name: app2-env1-alias
            port:
              number: 5002
      - path: /app2/env2
        pathType: Prefix
        backend:
          service:
            name: app2-env2-alias
            port:
              number: 5002
      - path: /app3
        pathType: Prefix
        backend:
          service:
            name: app3-env3-alias
            port:
              number: 5003
```

- using rewrite option to forward the request to the end point **/log**

4. testing

```
master@master:/usr/new/task2/ingress$ curl -X POST http://logservice.local:30343/app1/log -H "Content-Type: application/j
{"status":"logged"}
master@master:/usr/new/task2/ingress$ curl -X POST http://logservice.local:30343/app2A/log -H "Content-Type: application/
{"status":"logged"}
master@master:/usr/new/task2/ingress$ curl -X POST http://logservice.local:30343/app2B/log -H "Content-Type: application/
{"status":"logged"}
master@master:/usr/new/task2/ingress$ curl -X POST http://logservice.local:30343/app3/log -H "Content-Type: application/j
{"status":"logged"}
```

- from the results of snapshots, endpoints are reachable after using ingress
- logservice.local resolved to the master IP which is the ingress host
- ingress resolve paths to app1.env-1.svc.cluster.local

Step 5: Automation scripts

Deploy.sh

```
#!/bin/bash
set -e

VERSION=$(date +%Y%m%d%H%M)    # timestamp as tag


show_usage()
{
    echo "Deployment script V 1.0"
    echo "Usage: $0 [[--image=app1 or app2 or app3 ][--username= user personal docker hup
]  [--appsdir=directory of build] [--release= helm release name] [--namespace = env1]]"
}

show_help()
{
    show_usage
    echo "  where"
    echo "--image: the app image name e.g app1 , app2 , app3"
    echo "--usernam: Docker hub and the name tag of image"
    echo "--appsdir: Apps directory that have each app Docker folde e.g ./app1/Dockerfile"
    echo "--release: Release name of helm chart "
    echo "--namespace: namespace of app e.g env1 , env2 "

}
DEFAULTS() {
    DOCKER_USER=omargabr0
    DOCKER_DIR=/usr/new/task2/build-dir
    ENV=default
```

```
}

parse_args() {
    DEFAULTS
for i in "$@"; do
    case $i in

        --username=*) DOCKER_USER=${i#*=} ;;
        --appsdir=*) DOCKER_DIR=${i#*=} ;;
        --release=*)  RELEASE=${i#*=} ;;
        --image=*)  IMAGE_NAME=${i#*=} ;;
        --namespace=*) ENV=${i#*=} ;;
        --help)     show_help; exit 0 ;;
        *)          show_usage; exit 1 ;;
    esac

done
}
#login check
login(){
if docker info 2>/dev/null | grep -q "Username: $DOCKER_USER"; then
    echo " Already logged in to Docker Hub"
else
    echo " Not logged in. Login to docker hub first"
    exit 0
fi
}
#  Build image
build(){
echo ">>> Building Docker image..."
docker build -f $DOCKER_DIR/$IMAGE_NAME/Dockerfile.$IMAGE_NAME \
  -t $DOCKER_USER/$IMAGE_NAME:$VERSION \
  $DOCKER_DIR/$IMAGE_NAME
}

#  Push image
push(){
echo ">>> Pushing Docker image to Docker Hub..."
docker push $DOCKER_USER/$IMAGE_NAME:$VERSION
}

#  Update Kubernetes deployment
update() {
    if [ "$IMAGE_NAME" = "app2" ]; then
        helm upgrade $RELEASE ../$IMAGE_NAME/$IMAGE_NAME -f values-${ENV}.yaml --set
image.tag=$VERSION
    else
        echo ">>> Updating Kubernetes deployment Using Helm charts..."
```

```
        helm upgrade $RELEASE ../$IMAGE_NAME/$IMAGE_NAME --set image.tag=$VERSION
    fi
}

main (){

    parse_args "$@"
    login
    build
    push
    update



echo ">>> Deployment successful with image $IMAGE_NAME:$VERSION"
echo ">>> Helm shart $RELEASE updated with image $IMAGE_NAME:$VERSION"
}

main "$@"
```

- **Automation**: Builds, pushes, and deploys Docker images automatically.
- **Versioning**: Uses timestamp tags to ensure each image is unique.
- **Argument Handling**: Supports flexible inputs (image, user, release, namespace).
- **Defaults**: Provides fallback values if no arguments are given.
- **Login Check**: Ensures Docker Hub authentication before pushing.
- **Build & Push**: Creates Docker images from app Dockerfiles and pushes them to Docker Hub.
- **Deployment Update**: Uses Helm to update Kubernetes releases with the new image.
- **Reliability**: Script exits immediately on errors to avoid broken deployments.

Test 1: app1-env1

Rollout.sh

```
#!/bin/bash
set -e

show_usage() {
    echo "Rollback Script V1.0"
    echo "Usage: $0 [--release=helm_release] [--revision=number]"
```

```
}

DEFAULTS() {
    NAMESPACE=default
}

parse_args() {
    DEFAULTS
    for i in "$@"; do
        case $i in
            --release=*) RELEASE=${i#*=} ;;
            --namespace=*) NAMESPACE=${i#*=} ;;
            --revision=*) REVISION=${i#*=} ;;
            --help) show_usage; exit 0 ;;
            *) show_usage; exit 1 ;;
        esac
    done
}

rollback() {
    if [ -z "$RELEASE" ]; then
        echo "Error: --release is required"
        show_usage
        exit 1
    fi

    if [ -n "$REVISION" ]; then
        echo ">>> Rolling back Helm release $RELEASE in namespace $NAMESPACE to revision
$REVISION..."
        helm rollback $RELEASE $REVISION -n $NAMESPACE
    else
        echo ">>> Rolling back Helm release $RELEASE in namespace $NAMESPACE to previous
revision..."
        helm rollback $RELEASE -n $NAMESPACE
    fi
}

main() {
    parse_args "$@"
    rollback
    echo ">>> Rollback successful for release $RELEASE"
}

main "$@"
```