What do you mean by headless service
→ service which didnt have clusterip

Ingress :
  Ingress controller is app manage ingress rule
Use for path and host base routing

service account- communicate bet object
Helm

Kubectl get svc -A # check all
Kubectl get pod -n nginx-ingress
Kubectl describe pod podname
Kubectl get ingressClasses #to check class
—----------

usr/share/nginx/htm → nginx conf file path
usr/local/apache2/htdocs → httpd conf file path

Practical:

1) create dockerfile for laptop and build the img

```
FROM nginx:latest

# Create necessary directories
RUN mkdir -p /usr/share/nginx/html/laptop/

# Copy your index.html file
COPY index.html /usr/share/nginx/html/laptop/

# Expose port 80
EXPOSE 80

# Start the Nginx service
CMD ["nginx", "-g", "daemon off;"]
```

2) create dockerfile for mobile and build the img

```
FROM nginx:latest

# Create necessary directories
RUN mkdir -p /usr/share/nginx/html/laptop/

# Copy your index.html file
COPY index.html /usr/share/nginx/html/laptop/

# Expose port 80
EXPOSE 80

# Start the Nginx service
CMD ["nginx", "-g", "daemon off;"]
```

- **Push both mobile and laptop img to dockerhub**

3) Create deployment and service manifest for laptop and mobile

1. deployment and service yml file for laptop-app(laptop.yml)

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: laptop-deployment
  labels:
    app: laptop
spec:
  template:
    metadata:
      labels:
        app: laptop
    spec:
      containers:
        - name: laptop-c
          image: prathammore0025/top:v2
          ports:
            - containerPort: 80
  replicas: 3
  selector:
    matchLabels:
      app: laptop
  strategy:
    type: RollingUpdate
...
---
apiVersion: v1
kind: Service
metadata:
  name: laptop-svc
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app: laptop
...
```

2. **deployment and service yml file for mobile-app(mobile.yml)**

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mobile-deployment
  labels:
    app: mobile
spec:
  template:
    metadata:
      labels:
        app: mobile
    spec:
      containers:
        - name: mobile-c
          image: prathammore0025/mob:v1
          ports:
            - containerPort: 80
  replicas: 3
  selector:
    matchLabels:
      app: mobile
  strategy:
    type: RollingUpdate
...
---
apiVersion: v1
kind: Service
metadata:
  name: mobile-svc
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app: mobile
...
```

4) Apply both manifest mobile.yml and laptop.yml in cluster

5) install nginx-ingress controller in cluster (use nginx ref page)

6) set up load-balancer service in cluster (use nginx ref page)

7) set up ingress rule vai manifest create ingress.yml

1. **Ingress.yml**

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  defaultBackend:
    service:
      name: laptop-svc
      port:
        number: 80
  rules:
    - host:
"a8375bdfbd727419a84886318aaf60e4-979a36593be7f291.elb.eu-west-3.amazonaws.com"
      http:
        paths:
        - path: /mobile
          pathType: Prefix
          backend:
            service:
              name: mobile-svc
              port:
                number: 80
        - path: /laptop
          pathType: Prefix
          backend:
            service:
              name: laptop-svc
              port:
                number: 80
```

- **Change the DNS of load balancer in host**

- **Apply the ingress.yml in cluster**