Deploy nodejs Helm chart on AKS using GitHub Actions

Step #1:Create Azure AKS cluster using Aksctl:

Please follow below article to Create Azure AKS cluster using Aksctl.

Step #2:Create node.js Hello world application

In this step we need to create node.js code Hello world code using below code.

Create package.json file

package.json

--
-

```
{
  "name": "docker_web_app",
  "version": "1.0.0",
  "description": "Node.js on Docker",
  "author": "First Last <first.last@example.com>",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

==

Create server.js file

==

```
'use strict';
const express = require('express');
```

```
// Constants
const PORT = 8080;
const HOST = '0.0.0.0';
// App
const app = express();
app.get('/', (req, res) => {
  res.send('Hello World');
});
app.listen(PORT, HOST, () => {
  console.log(`Running on http://${HOST}:${PORT}`);
});
```

Create nodejs Dockerfile

Below is an example of a simple Dockerfile for a Node.js application:

```
# Use the official Node.js image as the base image
FROM node:18

# Set the working directory inside the container
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are copied
# where available (npm@5+)
# Copy package.json and package-lock.json to the working directory
COPY package*.json ./

# Install Node.js dependencies
RUN npm install
# If you are building your code for production
# RUN npm ci --omit=dev

# Bundle app source
COPY . .

# Expose the port on which your Node.js application will run
EXPOSE 8080

# Command to run your Node.js application
CMD [ "node", "server.js" ]
```
--

Create .dockerignore file

```
node_modules

npm-debug.log
```

Here’s a short and clear breakdown of the Dockerfile steps:

FROM node:18`** – Uses Node.js v18 as the base image.

WORKDIR /usr/src/app`** – Sets the working directory inside the container.

COPY package*.json ./`** – Copies dependency files.

RUN npm install`** – Installs Node.js dependencies.

COPY . .`** – Copies the rest of the app code.

EXPOSE 3000`** – Exposes port 3000 (used by the app).

CMD ["npm", "start"]`** – Runs the app using `npm start`.

Step #3:Build and run Node.js docker image

Note: Before run this command you need to install docker and give some permission

Run the following command

```
sudo apt  install docker.io

sudo usermod -aG docker $USER

sudo chmod 666 /var/run/docker.sock
```

To build the Docker image, navigate to the directory containing your Dockerfile and run the following command:

```
docker build . -t <your username>/node-app
```

Your image will now be listed by Docker:

```
docker images
```

Step #5:Push Node.js code and Docker on GitHub Repository:

Firstly let's clone your repo using below command:

```
git clone <your-repo-HTTPS>
```

Copy your node.js and docker files and paste in your repo folder

Then push this code into your repo using below commands:

```
git add .

git commit -m "files added"

git push
```

Step #6:Add Secrets in GitHub Repository

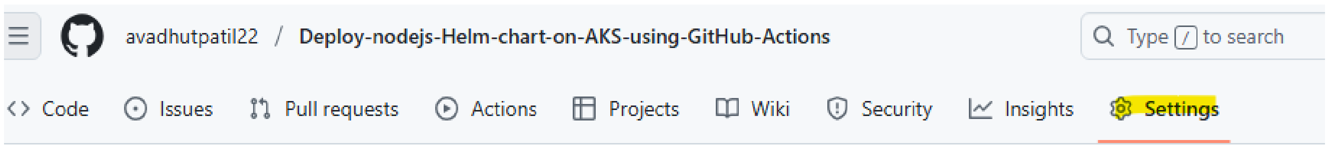
In our repository we need to add 1 secrets here:

run this given command on aks and you will get a key just add it in github secrets

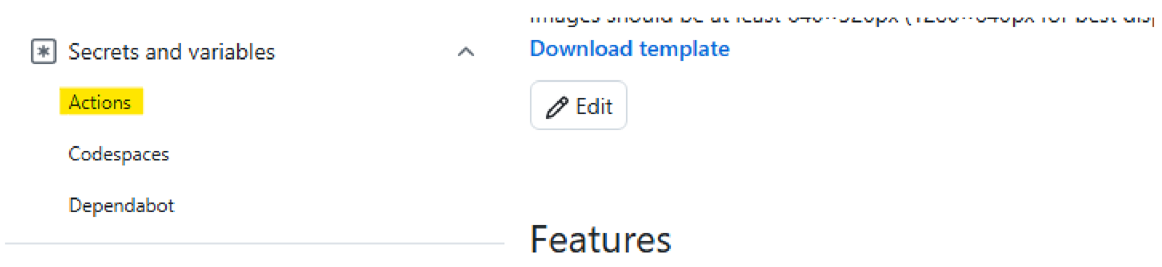
```
az ad sp create-for-rbac --name "github-spn" --role contributor \

  --scopes /subscriptions/<your-subscription-id> --sdk-auth
```

1) Steps to add secrets in github

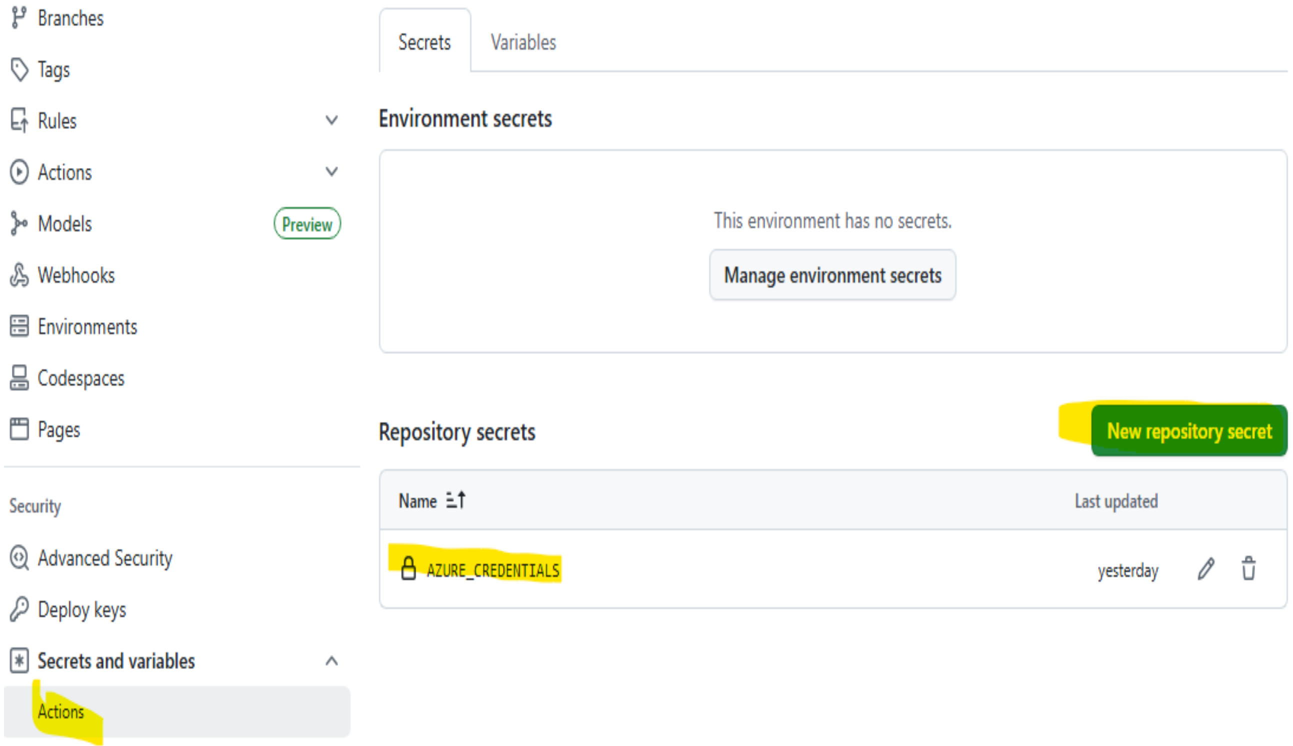
Go to your project settings



2) Go to actions



3) Create new secrets and past the output which you have get after run command.

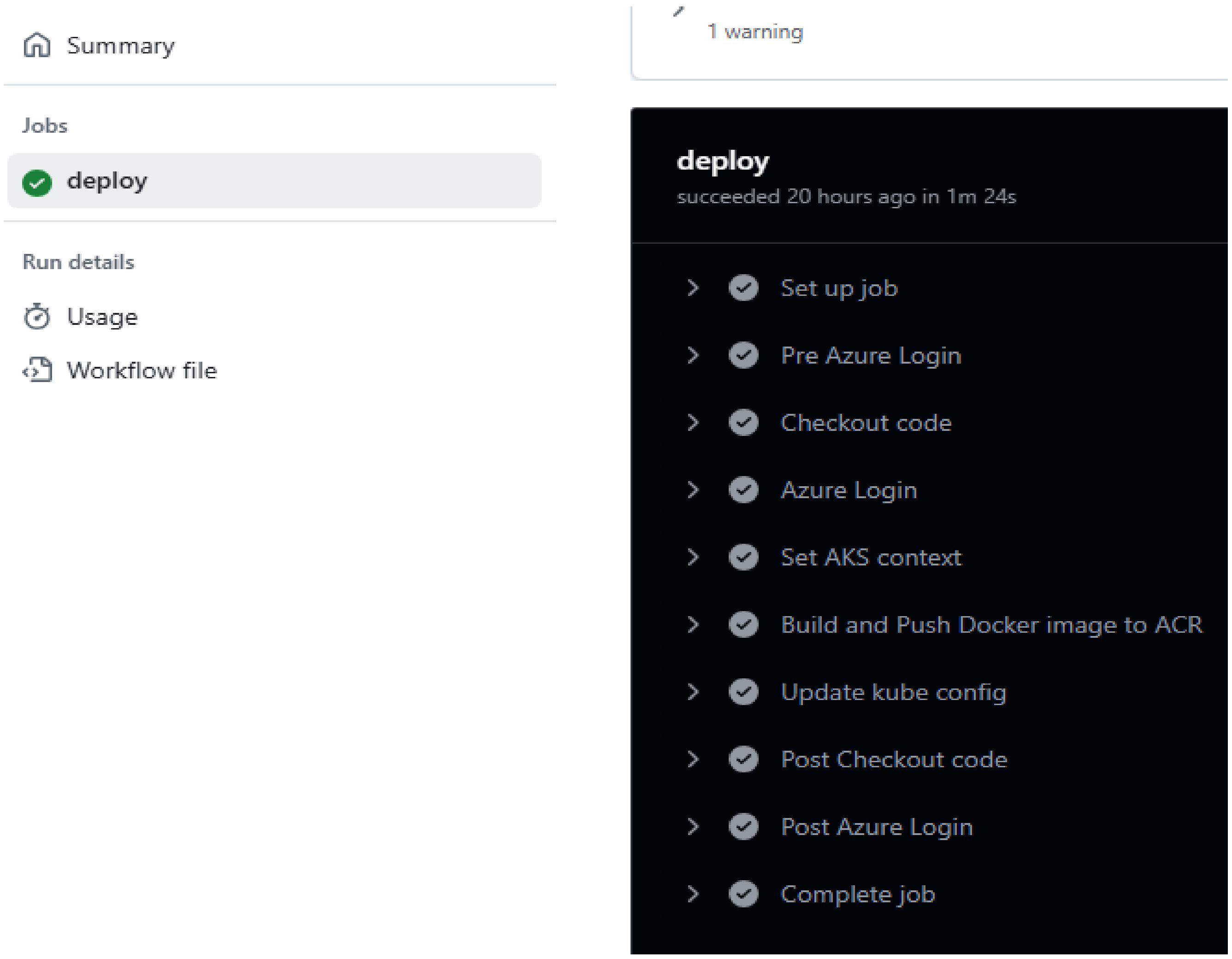


Name	Last updated
AZURE_CREDENTIALS	yesterday

Step #7:Create github Action workflow to build and push docker image to ACR

```
name: Node.js App Deploy to AKS

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Azure Login
      uses: azure/login@v1
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}

    - name: Set AKS context
      uses: azure/aks-set-context@v3
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}
        cluster-name: CKA
        resource-group: Avadhoot

    - name: Build and Push Docker image to ACR
      env:
        ACR_NAME: avadhoot
        IMAGE_NAME: avadhoot
        IMAGE_TAG: latest
      run: |
        az acr login --name $ACR_NAME
        docker build -t $ACR_NAME.azurecr.io/$IMAGE_NAME:$IMAGE_TAG .
        docker push $ACR_NAME.azurecr.io/$IMAGE_NAME:$IMAGE_TAG

    - name: Install Helm
      run: |
        curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

    - name: Deploy Helm chart
      run: |
        helm upgrade --install nodeapp ./node-app \
          --set image.repository=avadhoot.azurecr.io/avadhoot \
          --set image.tag=latest
```

Step #8:Run GitHub Action workflow


Summary
Jobs
deploy
Run details
Usage
Workflow file
1 warning

deploy
succeeded 20 hours ago in 1m 24s

- Set up job
- Pre Azure Login
- Checkout code
- Azure Login
- Set AKS context
- Build and Push Docker image to ACR
- Update kube config
- Post Checkout code
- Post Azure Login
- Complete job

Step #9:Install Helm on AKS cluster

Download the helm installation script using below command:

curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3

Add execute permissions to the downloaded script:

chmod +x get_helm.sh

Execute the installation script:

./get_helm.sh

Validate helm installation by executing the helm command:

helm

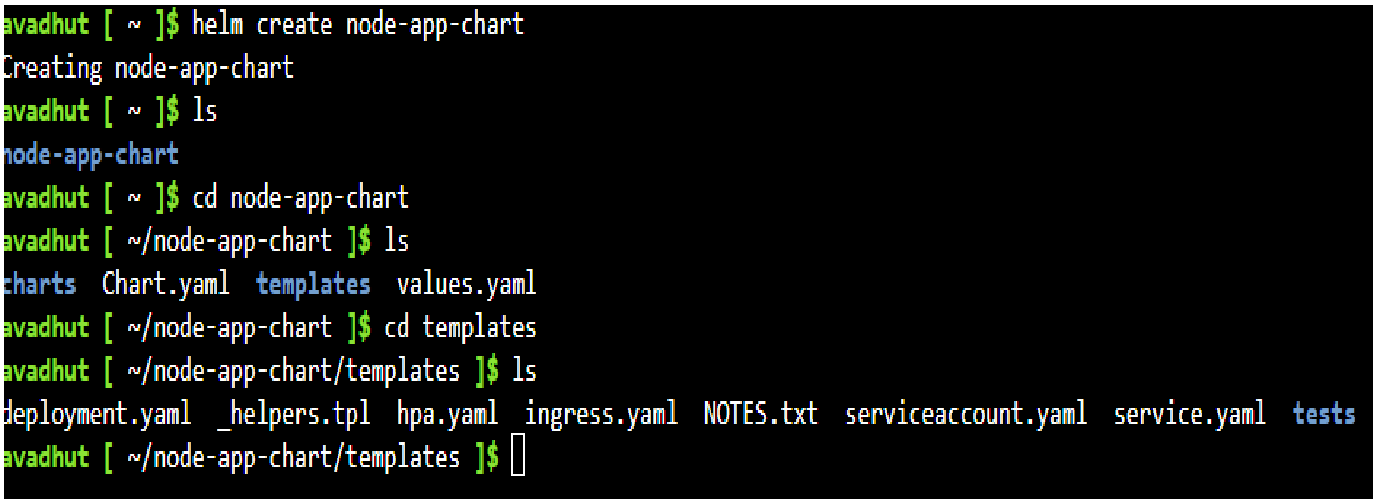
Check helm version:

helm version

Step #10:Create node-app helm chart

We will create a Helm chart node-app-chart for the Node.js application. To create the Helm chart, run this Helm command:

helm create node-app-chart



```
avadhut [ ~ ]$ helm create node-app-chart
Creating node-app-chart
avadhut [ ~ ]$ ls
node-app-chart
avadhut [ ~ ]$ cd node-app-chart
avadhut [ ~/node-app-chart ]$ ls
charts  Chart.yaml  templates  values.yaml
avadhut [ ~/node-app-chart ]$ cd templates
avadhut [ ~/node-app-chart/templates ]$ ls
deployment.yaml  _helpers.tpl  hpa.yaml  ingress.yaml  NOTES.txt  serviceaccount.yaml  service.yaml  tests
avadhut [ ~/node-app-chart/templates ]$ 
```

charts – Contains Helm chart dependencies (if any).

templates – Includes Kubernetes manifest files like deployment.yaml and service.yaml for app deployment.

Chart.yaml – Holds metadata about the Helm chart (name, version, etc.).

values.yaml – Defines config values (image, ports, replicas, service type) used in templates.

Step #11:modify helm chart files

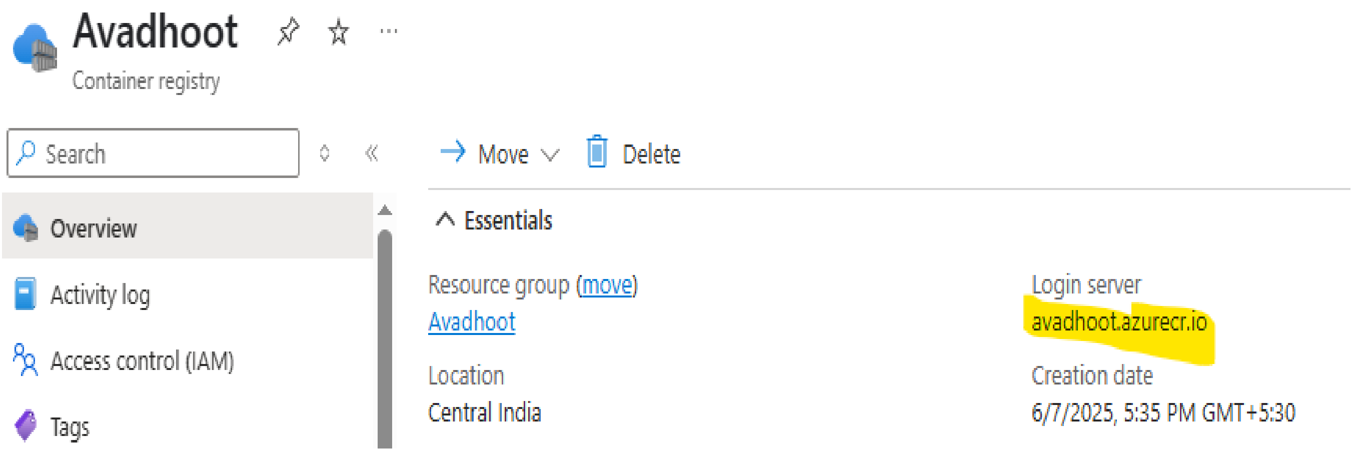
Lets modify values.yaml, deployment.yaml and service.yaml files using below code

values.yaml

```
# Default values for node-app-chart.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

replicaCount: 1

image:
  repository: avadhoot.azurecr.io
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  tag: "latest"

imagePullSecrets: []
nameOverride: ""
fullnameOverride: ""

serviceAccount:
  # Specifies whether a service account should be created
  create: true
  # Automatically mount a ServiceAccount's API credentials?
  automount: true
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name: ""

podAnnotations: {}
podLabels: {}

podSecurityContext: {}
  # fsGroup: 2000

securityContext: {}
  # capabilities:
```

```yaml
  #   drop:
  #   - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000

service:
  type: LoadBalancer
  port: 80
  targetPort: 8080
  protocol: TCP
  name: node-app

ingress:
  enabled: false
  className: ""
  annotations: {}
    # kubernetes.io/ingress.class: nginx
    # kubernetes.io/tls-acme: "true"
  hosts:
    - host: chart-example.local
      paths:
        - path: /
          pathType: ImplementationSpecific
  tls: []
  #  - secretName: chart-example-tls
  #    hosts:
  #      - chart-example.local

resources: {}
  # We usually recommend not to specify default resources and to leave this as a conscious
  # choice for the user. This also increases chances charts run on environments with little
  # resources, such as Minikube. If you do want to specify resources, uncomment the following
  # lines, adjust them as necessary, and remove the curly braces after 'resources:'.
  # limits:
  #   cpu: 100m
  #   memory: 128Mi
  # requests:
  #   cpu: 100m
  #   memory: 128Mi

livenessProbe:
  httpGet:
    path: /
    port: http
readinessProbe:
  httpGet:
    path: /
    port: http

autoscaling:
  enabled: false
  minReplicas: 1
  maxReplicas: 100
  targetCPUUtilizationPercentage: 80
  # targetMemoryUtilizationPercentage: 80

# Additional volumes on the output Deployment definition.
volumes: []
# - name: foo
#   secret:
#     secretName: mysecret
#     optional: false

# Additional volumeMounts on the output Deployment definition.
volumeMounts: []
# - name: foo
#   mountPath: "/etc/foo"
#   readOnly: true

nodeSelector: {}

tolerations: []

affinity: {}
```

As you can see in image repository section we need to paste ACR URI

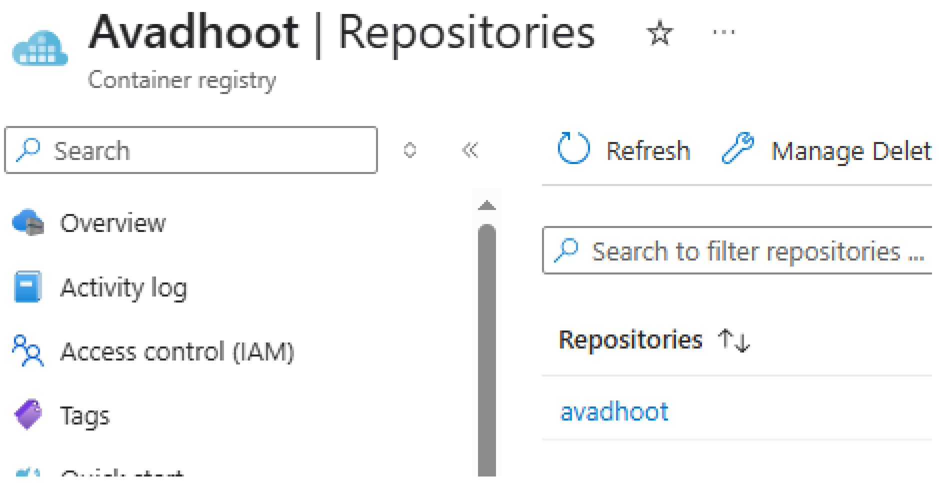


Resource group (move)
Avadhoot

Location
Central India

Login server
avadhoot.azurecr.io

Creation date
6/7/2025, 5:35 PM GMT+5:30



Repositories
avadhoot

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "node-app-chart.fullname" . }}
  labels:
    {{- include "node-app-chart.labels" . | nindent 4 }}
spec:
  {{- if not .Values.autoscaling.enabled }}
  replicas: {{ .Values.replicaCount }}
  {{- end }}
  selector:
    matchLabels:
      {{- include "node-app-chart.selectorLabels" . | nindent 6 }}
```

```
template:
  metadata:
    {{- with .Values.podAnnotations }}
    annotations:
      {{- toYaml . | nindent 8 }}
    {{- end }}
    labels:
      {{- include "node-app-chart.selectorLabels" . | nindent 8 }}
  spec:
    {{- with .Values.imagePullSecrets }}
    imagePullSecrets:
      {{- toYaml . | nindent 8 }}
    {{- end }}
    serviceAccountName: {{ include "node-app-chart.serviceAccountName" . }}
    securityContext:
      {{- toYaml .Values.podSecurityContext | nindent 8 }}
    containers:
      - name: {{ .Chart.Name }}
        securityContext:
          {{- toYaml .Values.securityContext | nindent 12 }}
        image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"
        imagePullPolicy: {{ .Values.image.pullPolicy }}
        ports:
          - name: http
            containerPort:  {{ .Values.service.targetPort }}
            protocol: TCP
        livenessProbe:
          httpGet:
            path: /
            port: http
        readinessProbe:
          httpGet:
            path: /
            port: http
        resources:
          {{- toYaml .Values.resources | nindent 12 }}
    {{- with .Values.nodeSelector }}
    nodeSelector:
      {{- toYaml . | nindent 8 }}
    {{- end }}
    {{- with .Values.affinity }}
    affinity:
      {{- toYaml . | nindent 8 }}
    {{- end }}
```

```
      {{- with .Values.tolerations }}
      tolerations:
        {{- toYaml . | nindent 8 }}
      {{- end }}
```
----

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: {{ include "node-app-chart.fullname" . }}
  labels:
    {{- include "node-app-chart.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: {{ .Values.service.targetPort }}
      protocol: {{ .Values.service.protocol }}
      name: {{ .Values.service.name }}
  selector:
    {{- include "node-app-chart.selectorLabels" . | nindent 4 }}
```

Step #12:Push node-app folder to GitHub repo
After modifying files then we need to push this node-app folder to GitHub repository

```
name: Node.js App Deploy to AKS
on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
```

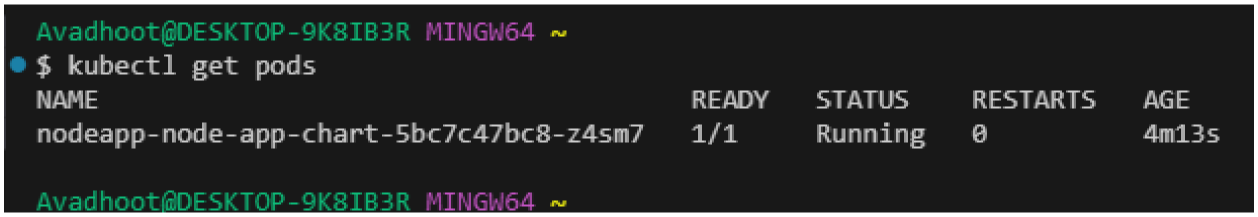
```
      uses: actions/checkout@v3

    - name: Azure Login
      uses: azure/login@v1
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}

    - name: Set AKS context
      uses: azure/aks-set-context@v3
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}
        cluster-name: CKA
        resource-group: Avadhoot

    - name: Build and Push Docker image to ACR
      env:
        ACR_NAME: avadhoot
        IMAGE_NAME: avadhoot
        IMAGE_TAG: latest
      run: |
        az acr login --name $ACR_NAME
        docker build -t $ACR_NAME.azurecr.io/$IMAGE_NAME:$IMAGE_TAG .
        docker push $ACR_NAME.azurecr.io/$IMAGE_NAME:$IMAGE_TAG

    - name: Install Helm
      run: |
        curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

    - name: Deploy Helm chart
      run: |
        helm upgrade --install nodeapp ./node-app \
          --set image.repository=avadhoot.azurecr.io/avadhoot \
          --set image.tag=latest
```

Step #14:Check pods, deployment and service on EKS

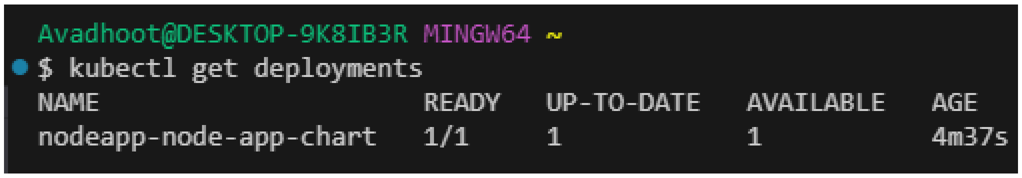
After successfully run our workflow lets check pods, deployment and service using below command:

```
kubectl get pods

kubectl get deployments

kubectl get service
```

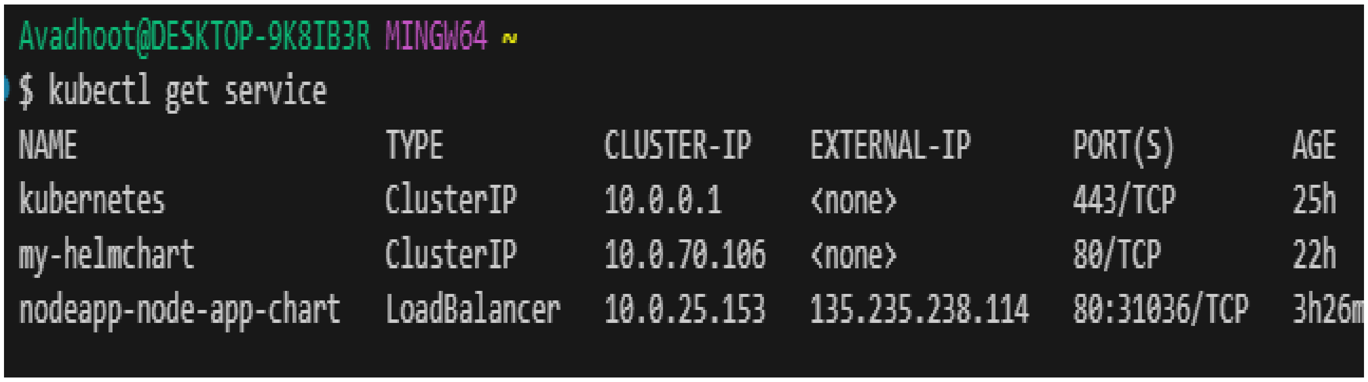
Output:



```
Avadhoot@DESKTOP-9K8IB3R MINGW64 ~
$ kubectl get pods
NAME                                      READY   STATUS    RESTARTS   AGE
nodeapp-node-app-chart-5bc7c47bc8-z4sm7   1/1     Running   0          4m13s

Avadhoot@DESKTOP-9K8IB3R MINGW64 ~
```



```
Avadhoot@DESKTOP-9K8IB3R MINGW64 ~
$ kubectl get deployments
NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
nodeapp-node-app-chart   1/1     1            1           4m37s
```



```
Avadhoot@DESKTOP-9K8IB3R MINGW64 ~
$ kubectl get service
NAME                     TYPE           CLUSTER-IP    EXTERNAL-IP       PORT(S)        AGE
kubernetes               ClusterIP      10.0.0.1      <none>            443/TCP        25h
my-helmchart             ClusterIP      10.0.70.106   <none>            80/TCP         22h
nodeapp-node-app-chart   LoadBalancer   10.0.25.153   135.235.238.114   80:31036/TCP   3h26m
```

----

if you will face imagepull back means you need to give permission use below commad

✅ Option 1: Attach ACR to AKS (Preferred)

Run this once only in CLI or Azure Cloud Shell:

```
az aks update \
  --name CKA \
  --resource-group Avadhoot \
  --attach-acr avadhoot
```

This automatically gives AKS permission to pull images from avadhoot.azurecr.io.