**Task - Dockerize and deploy a Node.js "Hello World" application on Kubernetes**

**1.Dockerize the Application:**

1. Clone the application repository.
2. Create a Dockerfile.
3. Build and push the Docker image to DockerHub.

**2.Prepare Kubernetes Deployment:**

1. Create a Helm chart for the application.
2. Deploy the application on a Kubernetes cluster using kubectl and Helm.

**3.Setup ArgoCD:**

1. Install ArgoCD.
2. Configure ArgoCD to deploy your Helm chart.

Here are the detailed steps for each part:

**1. Dockerize the Application**

1. Clone the Repository

Clone the Node.js "Hello World" application repository:  
*git clone https://github.com/johnpapa/node-hello.git*

*cd node-hello*

1. Create a Dockerfile

Create a Dockerfile in the root of the project:  
*# Stage 1: Build*

*FROM node:14 AS build*

*WORKDIR /app*

*COPY package\*.json ./*

*RUN npm install*

*COPY . .*

*# Stage 2: Run*

*FROM node:14-alpine*

*WORKDIR /app*

*COPY --from=build /app .*

*EXPOSE 3000*

*CMD ["node", "app.js"]*

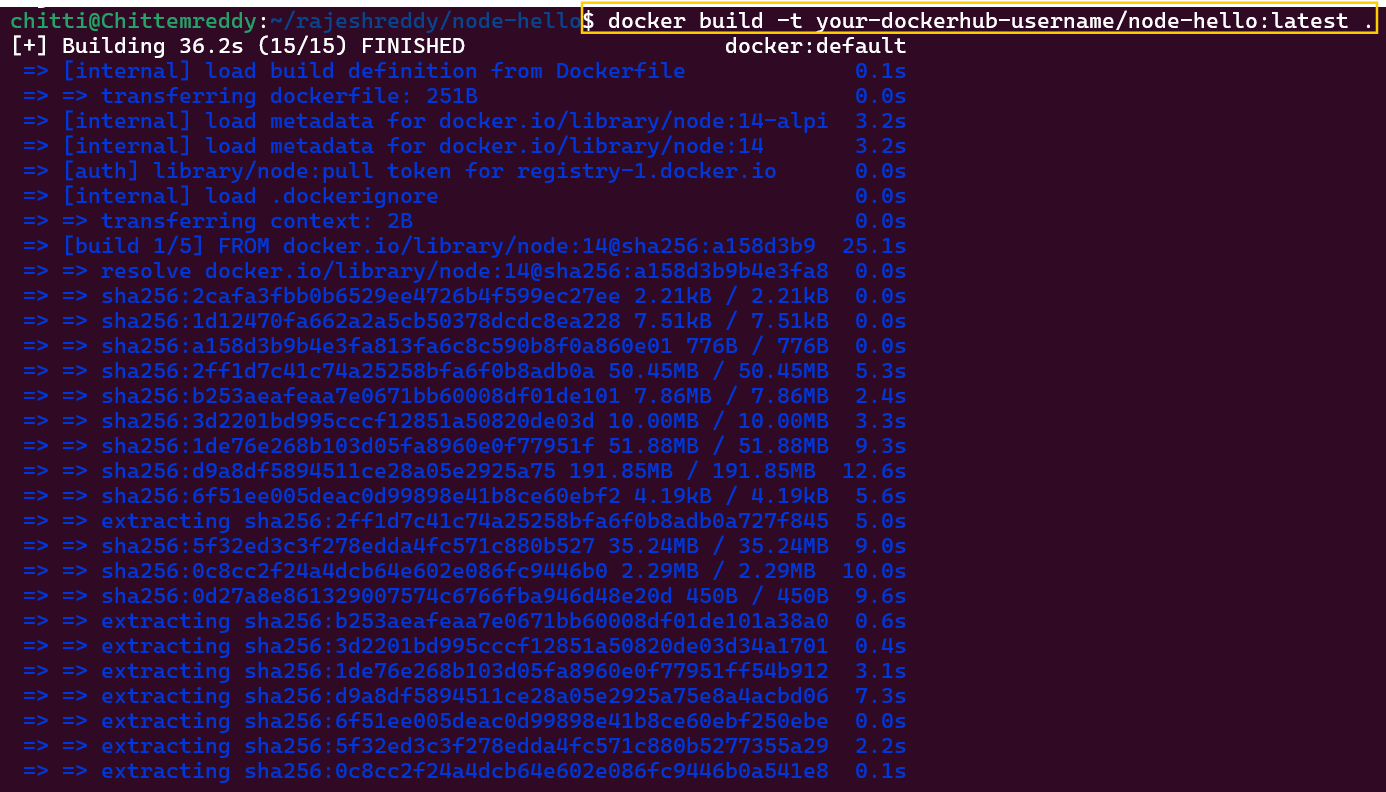
1. Build and Push Docker Image

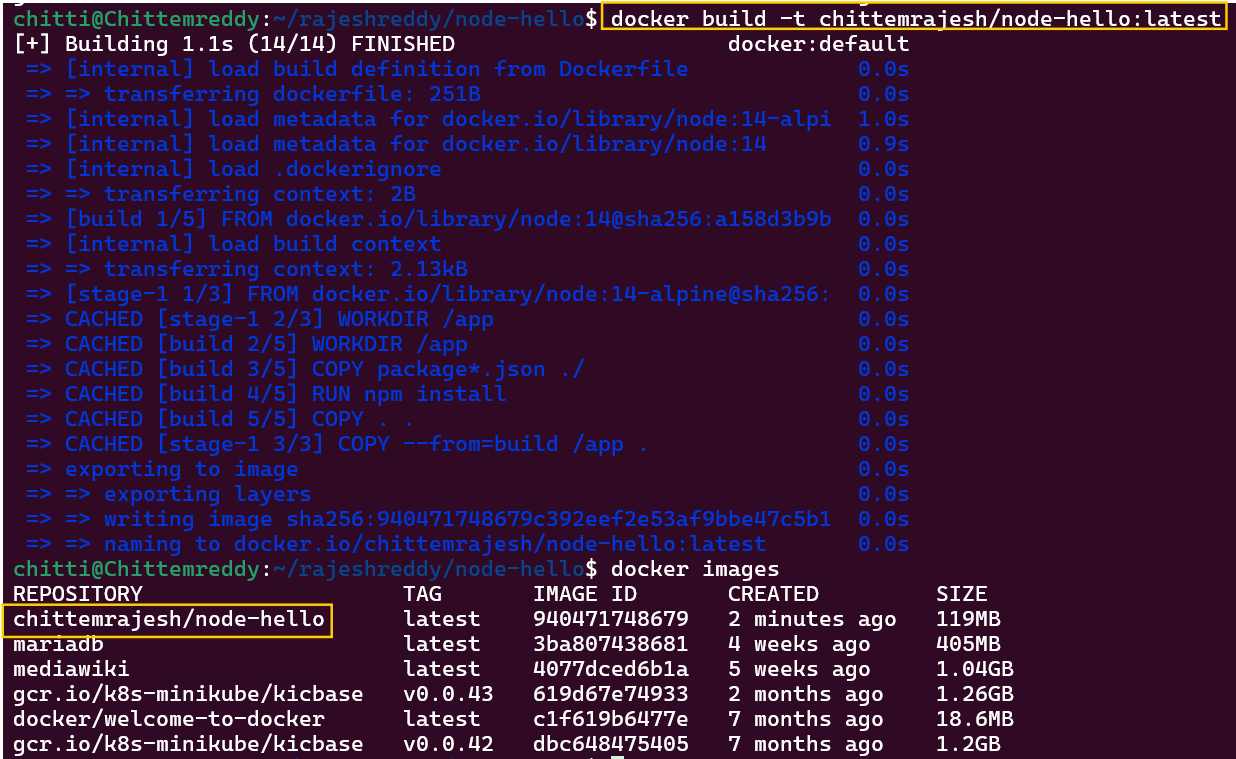
Build and push the Docker image to DockerHub*:  
# Log in to DockerHub*

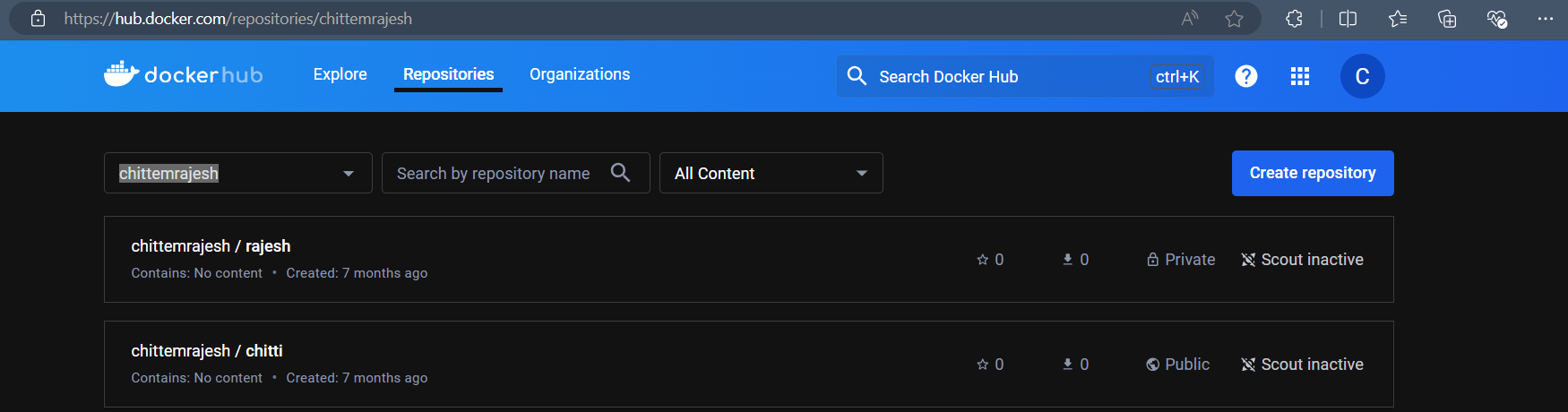
*docker login*

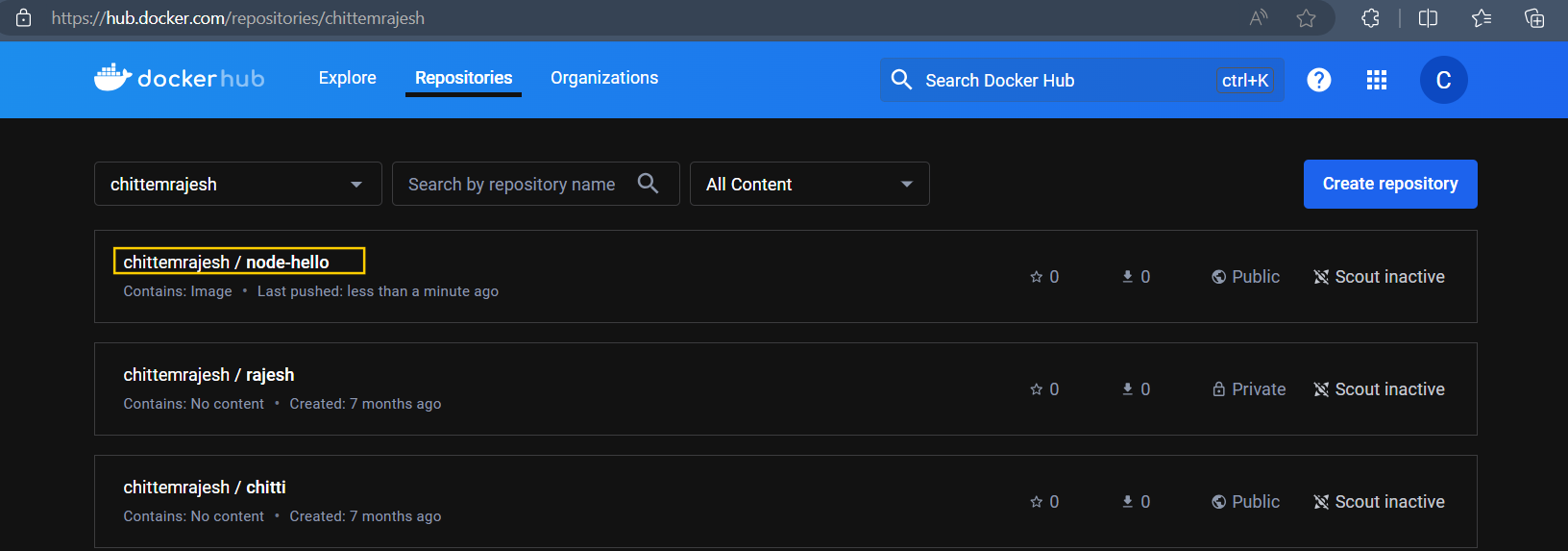
*# Build the Docker image*

*docker build -t your-dockerhub-username/node-hello:latest .*

**



*# Push the Docker image to DockerHub  
Before Push  
*

*After Push  
docker push chittemrajesh/node-hello:latest  
*

**2. Prepare Kubernetes Deployment**

1. Create a Helm Chart

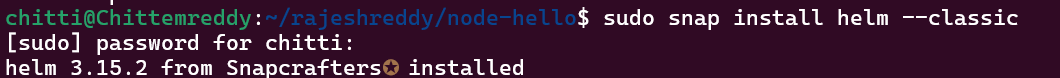
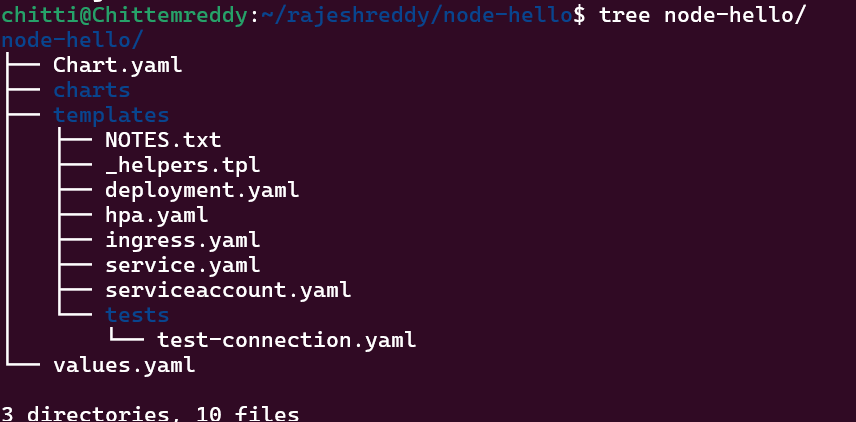
Create a Helm chart for the Node.js application. The structure will look like this:  
*node-hello/*

*Chart.yaml*

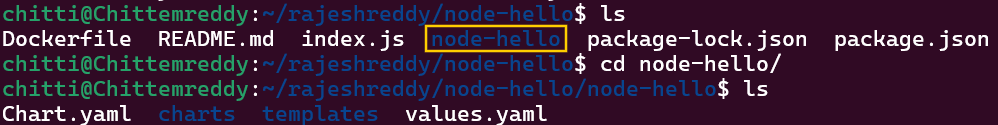
*values.yaml*

*templates/*

*deployment.yaml*

*service.yaml  
Installed helm* *Created Helm Chart  
  
View helm Chart  
*

*Goto Helm chart folder “node-hello” and View the all files*

**

*View the all files*

1. Chart.yaml

Cd node-hello

Vi Chart.yaml:  
*apiVersion: v2*

*name: node-hello*

*description: A Helm chart for Kubernetes*

*type: application*

*version: 0.1.0*

*appVersion: "1.16.0"*

1. values.yaml

Create node-hello/values.yaml:  
*# Default values for node-hello.*

*# This is a YAML-formatted file.*

*# Declare variables to be passed into your templates.*

*replicaCount: 1*

*image:*

*repository: chittemrajesh/node-hello*

*pullPolicy: IfNotPresent*

*# Overrides the image tag whose default is the chart appVersion.*

*tag: ""*

*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:*

*# drop:*

*# - ALL*

*# readOnlyRootFilesystem: true*

*# runAsNonRoot: true*

*# runAsUser: 1000*

*service:*

*type: ClusterIP*

*port: 8080*

*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: {}*

*# 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: []*

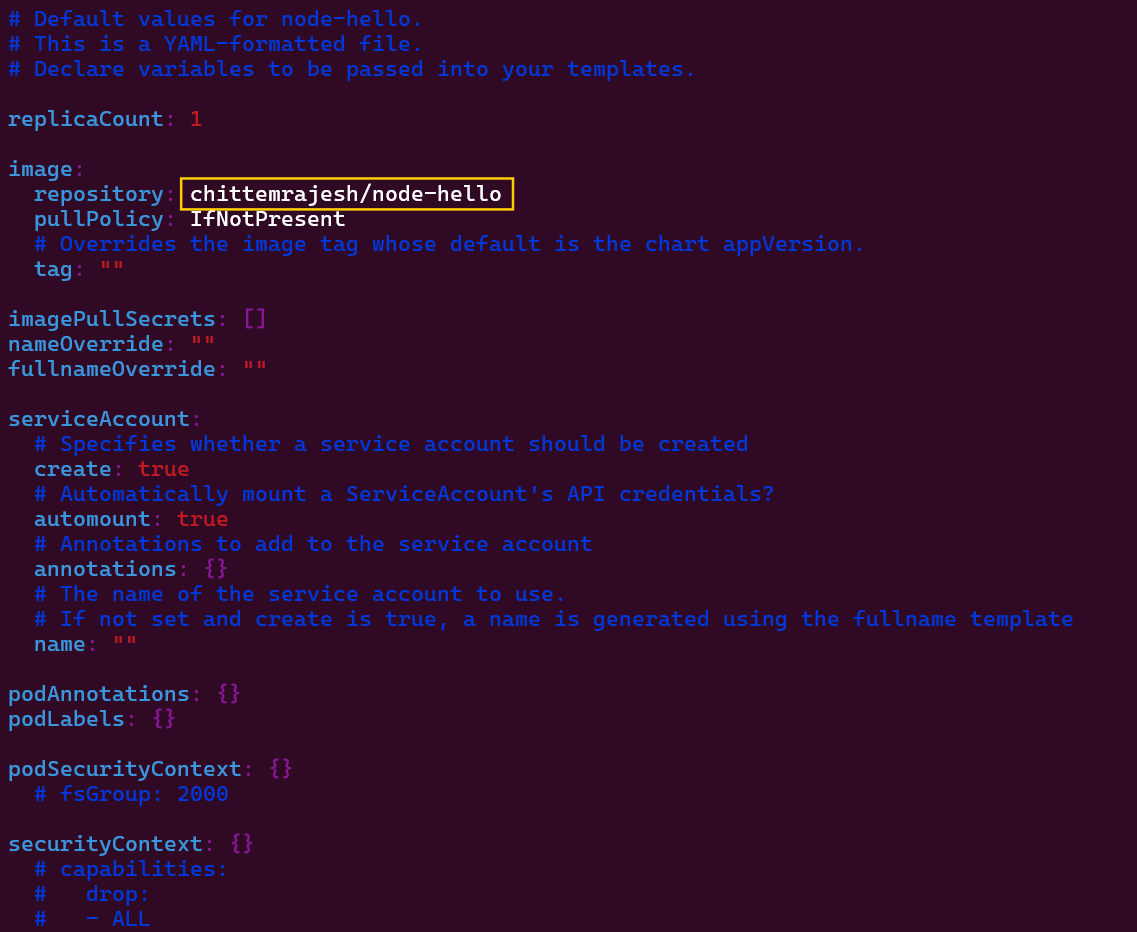
*# - name: foo*

*# mountPath: "/etc/foo"*

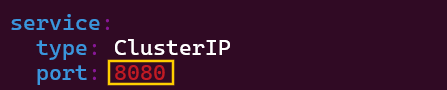
*# readOnly: true*

*nodeSelector: {}*

*tolerations: []*

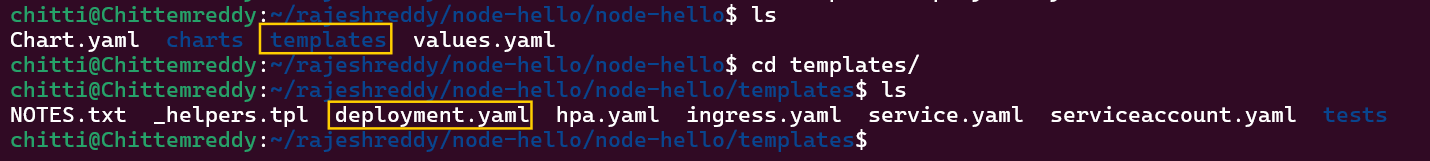
*affinity: {}  
change to our repository*

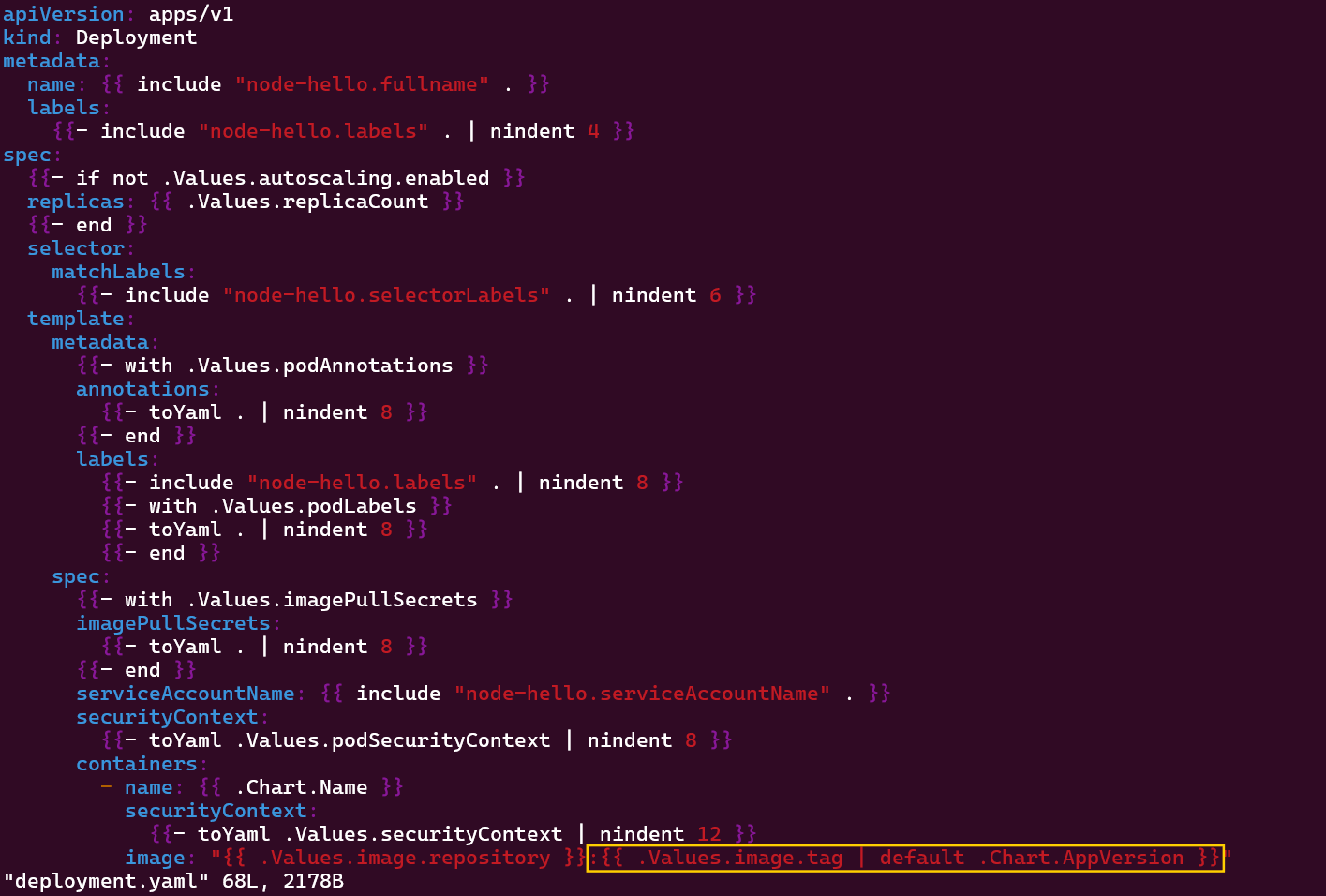
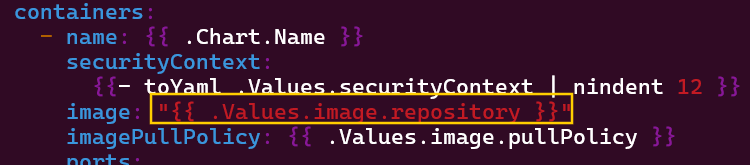
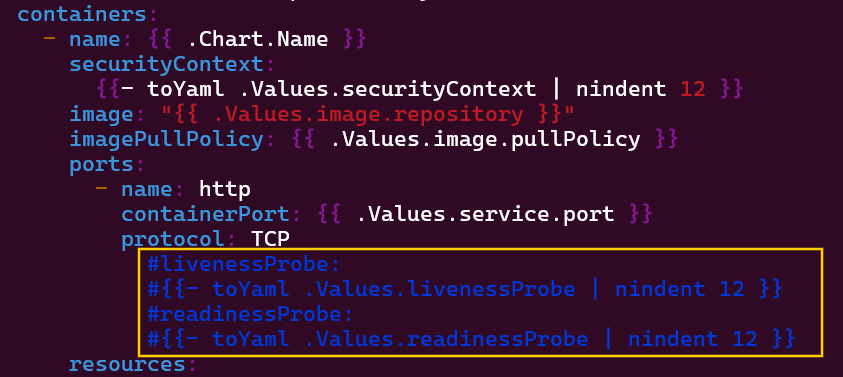
*change service port80 to 8080*

**

1. deployment.yaml

go template folder and ls



cd templates/deployment.yaml:  
  
  
*Commented*  
  
*apiVersion: apps/v1*

*kind: Deployment*

*metadata:*

*name: {{ include "node-hello.fullname" . }}*

*labels:*

*{{- include "node-hello.labels" . | nindent 4 }}*

*spec:*

*{{- if not .Values.autoscaling.enabled }}*

*replicas: {{ .Values.replicaCount }}*

*{{- end }}*

*selector:*

*matchLabels:*

*{{- include "node-hello.selectorLabels" . | nindent 6 }}*

*template:*

*metadata:*

*{{- with .Values.podAnnotations }}*

*annotations:*

*{{- toYaml . | nindent 8 }}*

*{{- end }}*

*labels:*

*{{- include "node-hello.labels" . | nindent 8 }}*

*{{- with .Values.podLabels }}*

*{{- toYaml . | nindent 8 }}*

*{{- end }}*

*spec:*

*{{- with .Values.imagePullSecrets }}*

*imagePullSecrets:*

*{{- toYaml . | nindent 8 }}*

*{{- end }}*

*serviceAccountName: {{ include "node-hello.serviceAccountName" . }}*

*securityContext:*

*{{- toYaml .Values.podSecurityContext | nindent 8 }}*

*containers:*

*- name: {{ .Chart.Name }}*

*securityContext:*

*{{- toYaml .Values.securityContext | nindent 12 }}*

*image: "{{ .Values.image.repository }}"*

*imagePullPolicy: {{ .Values.image.pullPolicy }}*

*ports:*

*- name: http*

*containerPort: {{ .Values.service.port }}*

*protocol: TCP*

*#livenessProbe:*

*#{{- toYaml .Values.livenessProbe | nindent 12 }}*

*#readinessProbe:*

*#{{- toYaml .Values.readinessProbe | nindent 12 }}*

*resources:*

*{{- toYaml .Values.resources | nindent 12 }}*

*{{- with .Values.volumeMounts }}*

*volumeMounts:*

*{{- toYaml . | nindent 12 }}*

*{{- end }}*

*{{- with .Values.volumes }}*

*volumes:*

*{{- toYaml . | nindent 8 }}*

*{{- end }}*

*{{- with .Values.nodeSelector }}*

*nodeSelector:*

*{{- toYaml . | nindent 8 }}*

*{{- end }}*

*{{- with .Values.affinity }}*

*affinity:*

*{{- toYaml . | nindent 8 }}*

*{{- end }}*

*{{- with .Values.tolerations }}*

*tolerations:*

*{{- toYaml . | nindent 8 }}*

*{{- end }}*

5. service.yaml  


Create node-hello/templates/service.yaml:  
*apiVersion: v1*

*kind: Service*

*metadata:*

*name: {{ include "node-hello.fullname" . }}*

*labels:*

*{{- include "node-hello.labels" . | nindent 4 }}*

*spec:*

*type: {{ .Values.service.type }}*

*ports:*

*- port: {{ .Values.service.port }}*

*targetPort: http*

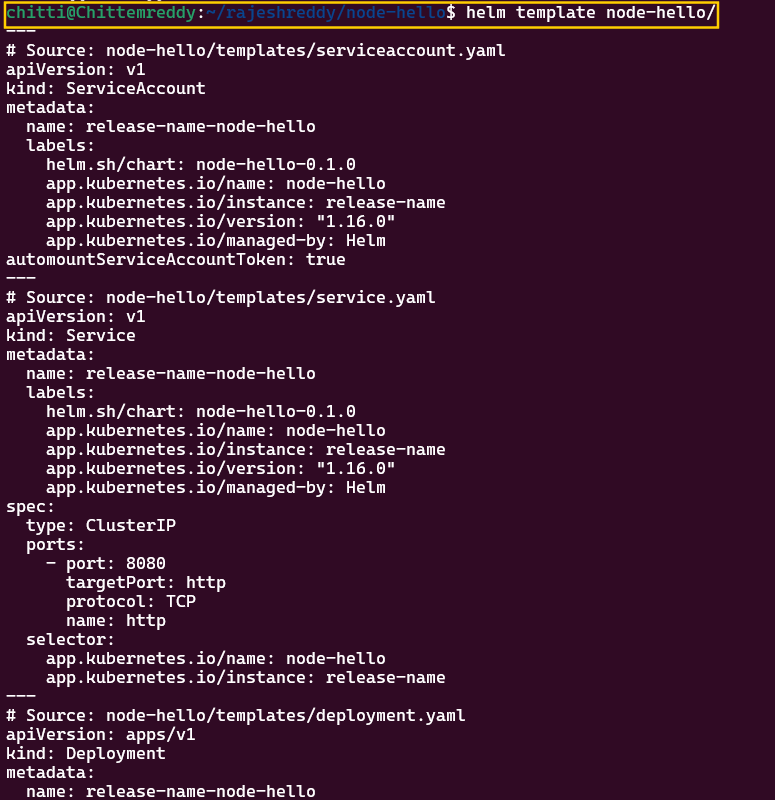
*protocol: TCP*

*name: http*

*selector:*

*{{- include "node-hello.selectorLabels" . | nindent 4 }}*

**4.See the Helm Package**helm template node-hello

*  
# Source: node-hello/templates/serviceaccount.yaml*

*apiVersion: v1*

*kind: ServiceAccount*

*metadata:*

*name: release-name-node-hello*

*labels:*

*helm.sh/chart: node-hello-0.1.0*

*app.kubernetes.io/name: node-hello*

*app.kubernetes.io/instance: release-name*

*app.kubernetes.io/version: "1.16.0"*

*app.kubernetes.io/managed-by: Helm*

*automountServiceAccountToken: true*

*---*

*# Source: node-hello/templates/service.yaml*

*apiVersion: v1*

*kind: Service*

*metadata:*

*name: release-name-node-hello*

*labels:*

*helm.sh/chart: node-hello-0.1.0*

*app.kubernetes.io/name: node-hello*

*app.kubernetes.io/instance: release-name*

*app.kubernetes.io/version: "1.16.0"*

*app.kubernetes.io/managed-by: Helm*

*spec:*

*type: ClusterIP*

*ports:*

*- port: 8080*

*targetPort: http*

*protocol: TCP*

*name: http*

*selector:*

*app.kubernetes.io/name: node-hello*

*app.kubernetes.io/instance: release-name*

*---*

*# Source: node-hello/templates/deployment.yaml*

*apiVersion: apps/v1*

*kind: Deployment*

*metadata:*

*name: release-name-node-hello*

*labels:*

*helm.sh/chart: node-hello-0.1.0*

*app.kubernetes.io/name: node-hello*

*app.kubernetes.io/instance: release-name*

*app.kubernetes.io/version: "1.16.0"*

*app.kubernetes.io/managed-by: Helm*

*spec:*

*replicas: 1*

*selector:*

*matchLabels:*

*app.kubernetes.io/name: node-hello*

*app.kubernetes.io/instance: release-name*

*template:*

*metadata:*

*labels:*

*helm.sh/chart: node-hello-0.1.0*

*app.kubernetes.io/name: node-hello*

*app.kubernetes.io/instance: release-name*

*app.kubernetes.io/version: "1.16.0"*

*app.kubernetes.io/managed-by: Helm*

*spec:*

*serviceAccountName: release-name-node-hello*

*securityContext:*

*{}*

*containers:*

*- name: node-hello*

*securityContext:*

*{}*

*image: "chittemrajesh/node-hello"*

*imagePullPolicy: IfNotPresent*

*ports:*

*- name: http*

*containerPort: 8080*

*protocol: TCP*

*livenessProbe:*

*httpGet:*

*path: /*

*port: http*

*readinessProbe:*

*httpGet:*

*path: /*

*port: http*

*resources:*

*{}*

*---*

*# Source: node-hello/templates/tests/test-connection.yaml*

*apiVersion: v1*

*kind: Pod*

*metadata:*

*name: "release-name-node-hello-test-connection"*

*labels:*

*helm.sh/chart: node-hello-0.1.0*

*app.kubernetes.io/name: node-hello*

*app.kubernetes.io/instance: release-name*

*app.kubernetes.io/version: "1.16.0"*

*app.kubernetes.io/managed-by: Helm*

*annotations:*

*"helm.sh/hook": test*

*spec:*

*containers:*

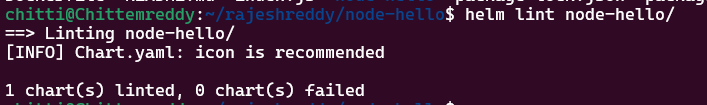
*- name: wget*

*image: busybox*

*command: ['wget']*

*args: ['release-name-node-hello:8080']*

*restartPolicy: Never  
see the 1 chart going to update*

**

1. **helm -debug -dry-run**

Deploy the Helm Chart

Deploy the Helm chart to your Kubernetes cluster:  
*helm install release-name-node-hello --debug --dry-run node-hello*

1. **Verify the helm install**

*helm list -a*

1. **Verify**

kubectl get all

IMP!  
take here service loadbalancer Ip address and paste in to the browser will get the node hello world code

Note: I am not able to paste screen short from here! Due my ubuntu getting trouble, not able the get the pods, service, deployment, namespace details, due to below the issue, I have capable to do that, But I really sorry by casing this issue  


**5. Setup ArgoCD**

1. **Install ArgoCD**

Install ArgoCD on your Kubernetes cluster:  
*kubectl create namespace node-hello-n*

*kubectl apply -n node-hello-n -f deployment.yaml*

1. **Configure ArgoCD CLI**

Install the ArgoCD CLI and login:  
*brew install argocd # For macOS users, otherwise check ArgoCD CLI installation instructions*

*# Forward ArgoCD server port*

*kubectl port-forward svc/argocd-server -n node-hello-n 8080:443*

*# Login to ArgoCD*

*argocd login localhost:8080*

1. **ArgoCD Application Configuration**

Create argocd/node-hello-app.yaml:  
*apiVersion: argoproj.io/v1alpha1*

*kind: Application*

*metadata:*

*name: node-hello*

*namespace: node-hello-n*

*spec:*

*project: default*

*source:*

*repoURL: “https://github.com/chittemrajesh2/Node-Hello-Docker-K8-Helm-ArgoCD.git”*

*targetRevision: HEAD*

*path: helm/node-hello*

*destination:*

*server: 'https://kubernetes.default.svc'*

*namespace: default*

*syncPolicy:*

*automated:*

*prune: true*

*selfHeal: true*

1. **Apply the ArgoCD Application**

Apply the ArgoCD application configuration:  
*kubectl apply -f argocd/*node-hello *-app.yaml*

**5.Verify**

kubectl get all

IMP!  
take here service loadbalancer Ip address and paste in to the browser will get the “Argo CD” and give user name and password,

And configure argo cd and when next time when will update the yaml or add, automatically sync pods, deployment, services

Note: to get the password :

*kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d*

**Conclusion**:

* It’ll also be presented in the Argo web UI as well
* The app shows as Missing and Out of Sync. So far, we’ve only registered the app with Argo; a separate sync operation is required to actually create the resources in our Kubernetes cluster. We can Press the Sync or Sync Apps button in the UI, or use the argocd app sync command, to start your initial sync.
* *argocd app sync node-hello*
* By Clicking the app’s card in the web UI allows you to easily inspect the components in the chart. We can also see the relationships between them and take action to force a resync or rollback. These operations can also be achieved using the CLI.
* We can now update your deployment by modifying **your Helm chart, pushing the changes to GitHub, and initiating a resync within Argo.**