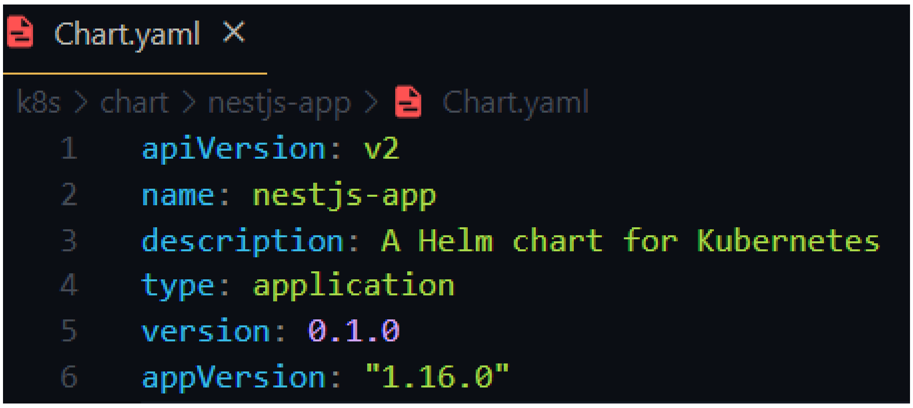
Desafío 12

El primer paso para este desafío es crear dentro de la carpeta k8s una carpeta llamada **Chart.yaml**

El archivo **Chart.yaml** es donde se definen los metadatos principales de un **Helm chart**, como el nombre, la versión, y una breve descripción del chart.

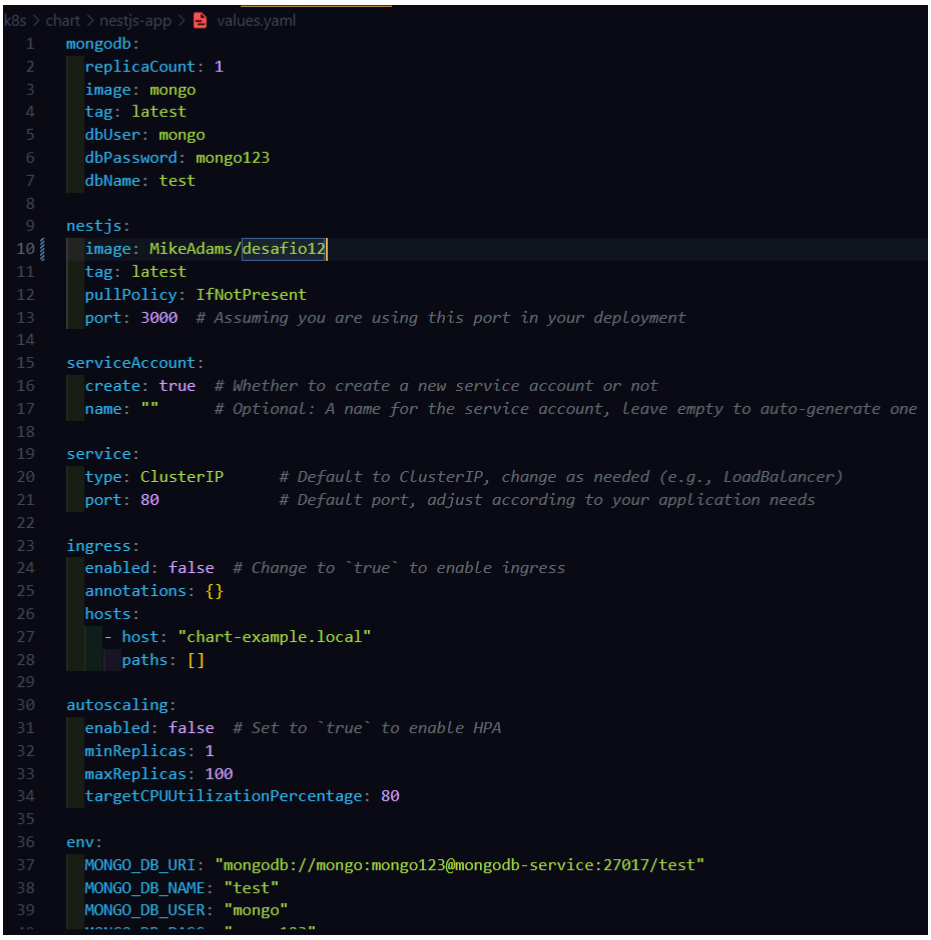


```yaml
apiVersion: v2
name: nestjs-app
description: A Helm chart for Kubernetes
type: application
version: 0.1.0
appVersion: "1.16.0"
```

seguimos con el archivo **values.yaml** dentro de la misma carpeta de nuestro chart

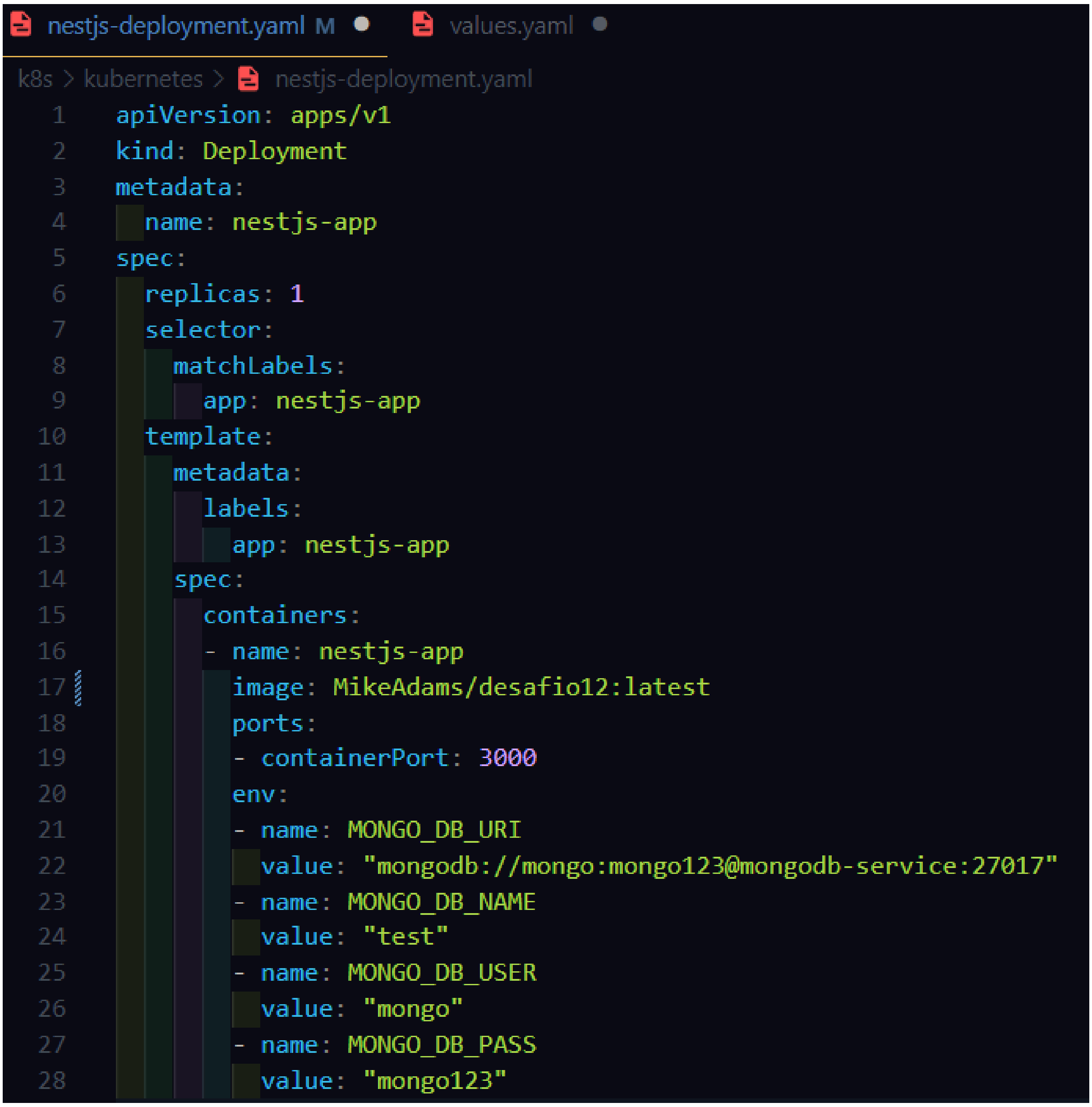
el archivo **values.yam**

El archivo **values.yam** define los valores por defecto para las variables de configuración de un **Helm chart**, permitiendo personalizar el despliegue sin modificar los templates directamente.



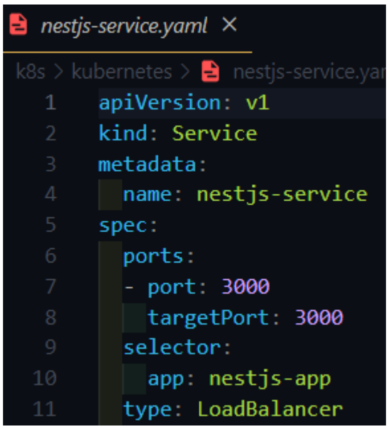
```yaml
mongodb:
  replicaCount: 1
  image: mongo
  tag: latest
  dbUser: mongo
  dbPassword: mongo123
  dbName: test

nestjs:
  image: MikeAdams/desafio12
  tag: latest
  pullPolicy: IfNotPresent
  port: 3000  # Assuming you are using this port in your deployment

serviceAccount:
  create: true  # Whether to create a new service account or not
  name: ""      # Optional: A name for the service account, leave empty to auto-generate one

service:
  type: ClusterIP      # Default to ClusterIP, change as needed (e.g., LoadBalancer)
  port: 80             # Default port, adjust according to your application needs

ingress:
  enabled: false  # Change to `true` to enable ingress
  annotations: {}
  hosts:
    - host: "chart-example.local"
      paths: []

autoscaling:
  enabled: false  # Set to `true` to enable HPA
  minReplicas: 1
  maxReplicas: 100
  targetCPUUtilizationPercentage: 80

env:
  MONGO_DB_URI: "mongodb://mongo:mongo123@mongodb-service:27017/test"
  MONGO_DB_NAME: "test"
  MONGO_DB_USER: "mongo"
```

Como en el desafío anterior crearemos nuestros archivos ya conocidos
nestjs-deployment.yaml


nestjs-deployment.yaml M ●
values.yaml ●
k8s > kubernetes > nestjs-deployment.yaml

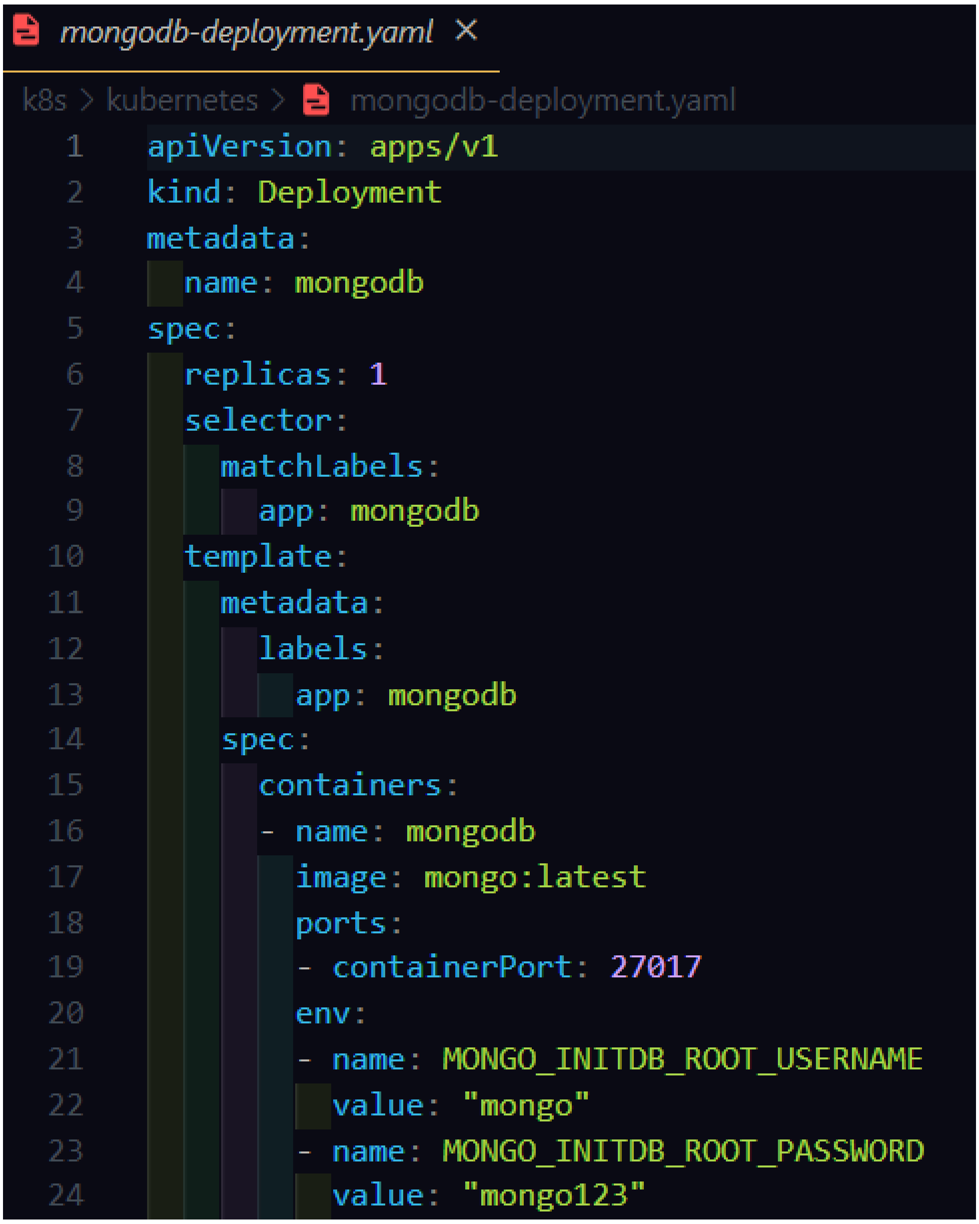
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nestjs-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nestjs-app
  template:
    metadata:
      labels:
        app: nestjs-app
    spec:
      containers:
      - name: nestjs-app
        image: MikeAdams/desafio12:latest
        ports:
        - containerPort: 3000
        env:
        - name: MONGO_DB_URI
          value: "mongodb://mongo:mongo123@mongodb-service:27017"
        - name: MONGO_DB_NAME
          value: "test"
        - name: MONGO_DB_USER
          value: "mongo"
        - name: MONGO_DB_PASS
          value: "mongo123"
```

nestjs-service.yaml



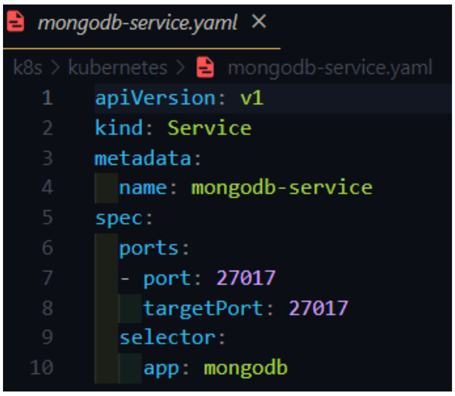
```yaml
apiVersion: v1
kind: Service
metadata:
  name: nestjs-service
spec:
  ports:
  - port: 3000
    targetPort: 3000
  selector:
    app: nestjs-app
  type: LoadBalancer
```

El archivo **mongodb-deployment.yaml** que es un manifiesto de **Kubernetes** que se utiliza para gestionar el despliegue de **MongoDB** en un clúster de **Kubernetes**.


mongodb-deployment.yaml ×
k8s > kubernetes > mongodb-deployment.yaml

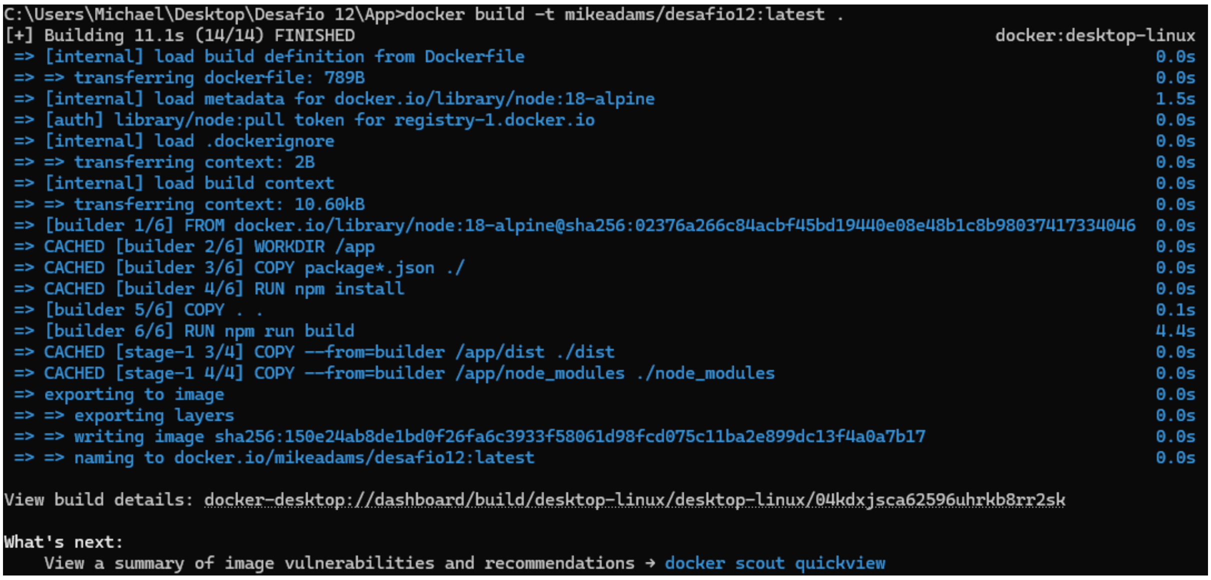
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongodb
        image: mongo:latest
        ports:
        - containerPort: 27017
        env:
        - name: MONGO_INITDB_ROOT_USERNAME
          value: "mongo"
        - name: MONGO_INITDB_ROOT_PASSWORD
          value: "mongo123"
```

El archivo **mongodb-service** es un manifiesto de **Kubernetes** que define un Service para **MongoDB**.



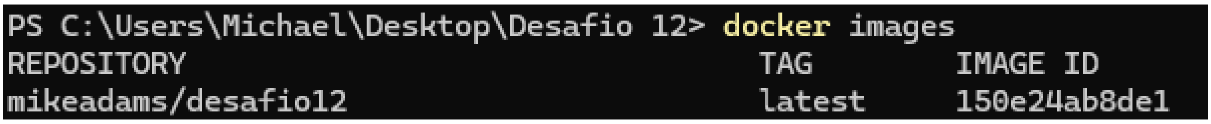
```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  ports:
  - port: 27017
    targetPort: 27017
  selector:
    app: mongodb
```

Corremos el comando en consola de powershell **docker build -t <Nombre Imagen>**

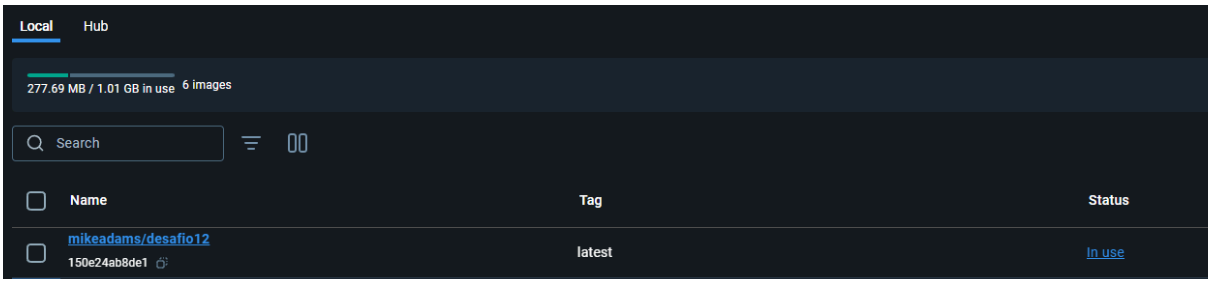


```
C:\Users\Michael\Desktop\Desafio 12\App>docker build -t mikeadams/desafio12:latest .
[+] Building 11.1s (14/14) FINISHED                                                       docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                      0.0s
 => => transferring dockerfile: 789B                                                                      0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                                         1.5s
 => [auth] library/node:pull token for registry-1.docker.io                                               0.0s
 => [internal] load .dockerignore                                                                         0.0s
 => => transferring context: 2B                                                                           0.0s
 => [internal] load build context                                                                         0.0s
 => => transferring context: 10.60kB                                                                      0.0s
 => [builder 1/6] FROM docker.io/library/node:18-alpine@sha256:02376a266c84acbf45bd19440e08e48b1c8b98037417334046  0.0s
 => CACHED [builder 2/6] WORKDIR /app                                                                     0.0s
 => CACHED [builder 3/6] COPY package*.json ./                                                            0.0s
 => CACHED [builder 4/6] RUN npm install                                                                  0.0s
 => [builder 5/6] COPY . .                                                                                0.1s
 => [builder 6/6] RUN npm run build                                                                       4.4s
 => CACHED [stage-1 3/4] COPY --from=builder /app/dist ./dist                                             0.0s
 => CACHED [stage-1 4/4] COPY --from=builder /app/node_modules ./node_modules                             0.0s
 => exporting to image                                                                                    0.0s
 => => exporting layers                                                                                   0.0s
 => => writing image sha256:150e24ab8de1bd0f26fa6c3933f58061d98fcd075c11ba2e899dc13f4a0a7b17               0.0s
 => => naming to docker.io/mikeadams/desafio12:latest                                                     0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/04kdxjsca62596uhrkb8rr2sk

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

Vemos la imagen que hemos creado



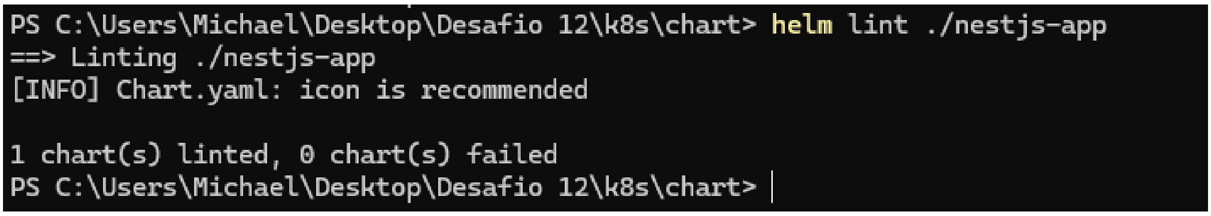
```
PS C:\Users\Michael\Desktop\Desafio 12> docker images
REPOSITORY                                  TAG         IMAGE ID
mikeadams/desafio12                         latest      150e24ab8de1
```



277.69 MB / 1.01 GB in use 6 images

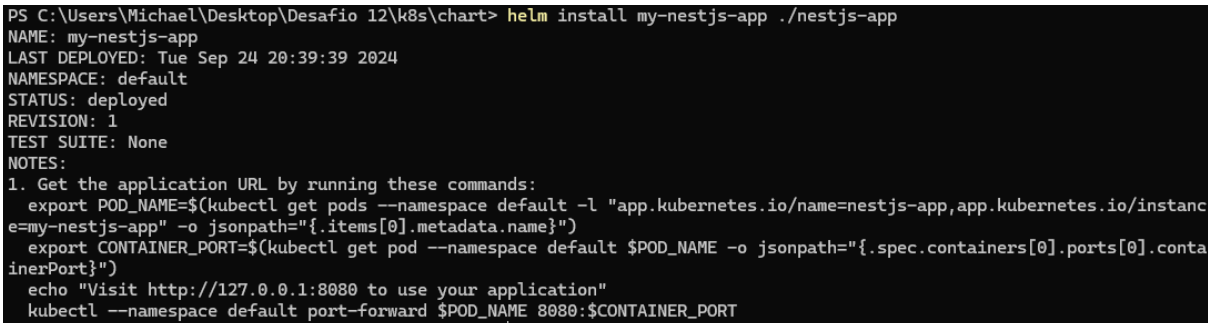
□	Name	Tag	Status
□	mikeadams/desafio12 150e24ab8de1	latest	In use

Con el comando **helm lint ./nestjs-app** verificamos que el chart de Helm ubicado en el directorio **./nestjs-app** esté bien configurado



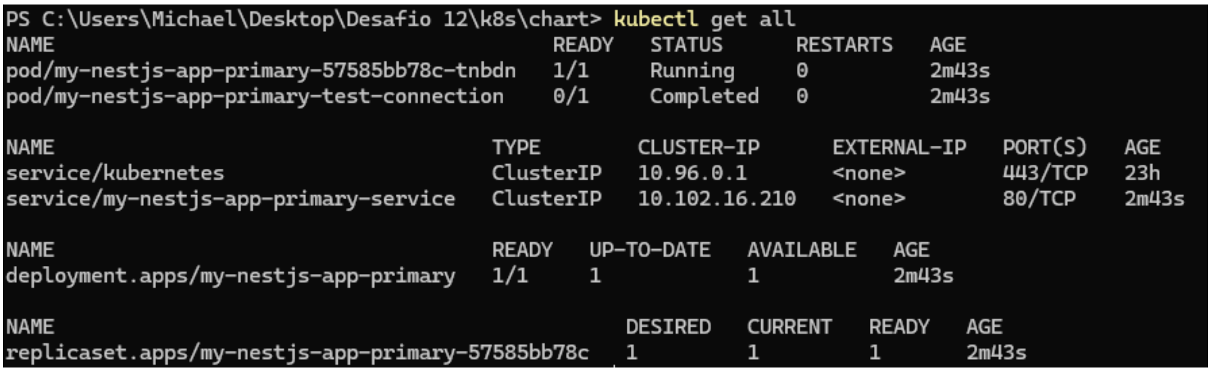
```
PS C:\Users\Michael\Desktop\Desafio 12\k8s\chart> helm lint ./nestjs-app
==> Linting ./nestjs-app
[INFO] Chart.yaml: icon is recommended

1 chart(s) linted, 0 chart(s) failed
PS C:\Users\Michael\Desktop\Desafio 12\k8s\chart> |
```

Ya por último corremos el comando **helm install my-nestjs-app ./nestjs-app** y el nos desplegará nuestra app **kubernetes**



```
PS C:\Users\Michael\Desktop\Desafio 12\k8s\chart> helm install my-nestjs-app ./nestjs-app
NAME: my-nestjs-app
LAST DEPLOYED: Tue Sep 24 20:39:39 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=nestjs-app,app.kubernetes.io/instanc
e=my-nestjs-app" -o jsonpath="{.items[0].metadata.name}")
  export CONTAINER_PORT=$(kubectl get pod --namespace default $POD_NAME -o jsonpath="{.spec.containers[0].ports[0].conta
inerPort}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace default port-forward $POD_NAME 8080:$CONTAINER_PORT
```

Hacemos un **kubctl get all** para fijarnos el estado de los recursos de nuestro cluster.



```
PS C:\Users\Michael\Desktop\Desafio 12\k8s\chart> kubectl get all
NAME                                        READY   STATUS      RESTARTS   AGE
pod/my-nestjs-app-primary-57585bb78c-tnbdn  1/1     Running     0          2m43s
pod/my-nestjs-app-primary-test-connection   0/1     Completed   0          2m43s

NAME                                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/kubernetes                      ClusterIP   10.96.0.1       <none>        443/TCP   23h
service/my-nestjs-app-primary-service   ClusterIP   10.102.16.210   <none>        80/TCP    2m43s

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-nestjs-app-primary   1/1     1            1           2m43s

NAME                                               DESIRED   CURRENT   READY   AGE
replicaset.apps/my-nestjs-app-primary-57585bb78c   1         1         1       2m43s
```