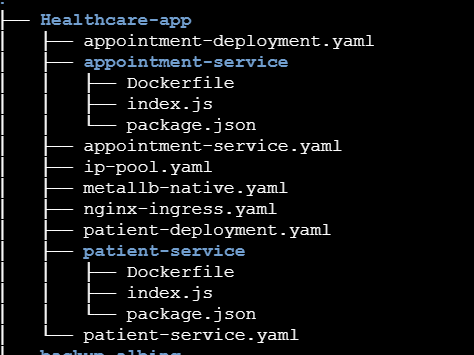
**Two healthcare-micro services-deployed-with-minikube**

**Summary:**

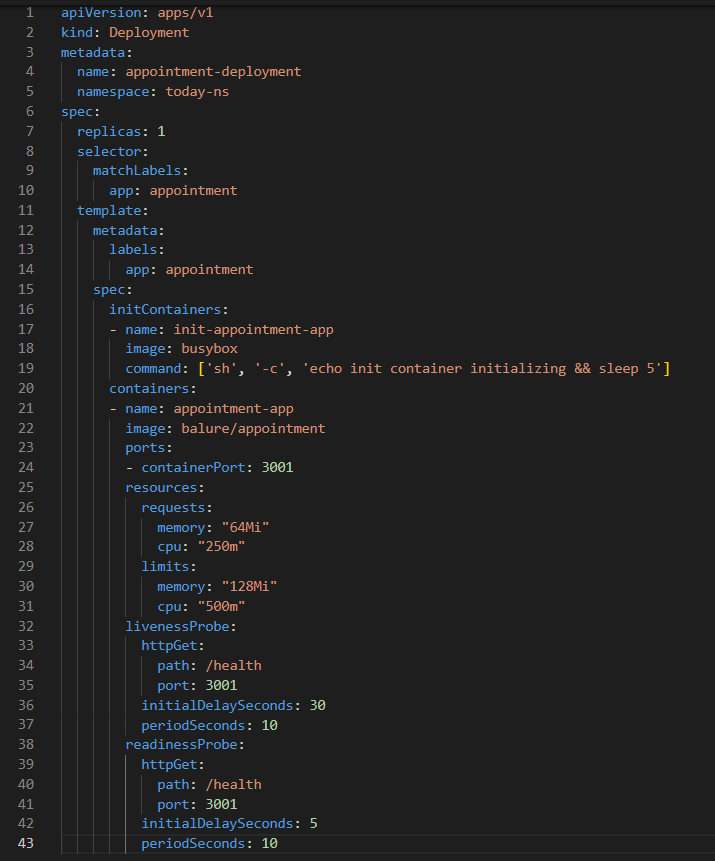
This project deploys a **healthcare application** consisting of two micro services **Appointment Service** and **Patient Service** on a **Minikube** Kubernetes cluster. Both services are exposed using **Load Balancer** type services and managed through **Ingress** for path-based routing

* **Healthcare.in/appointment** for the Appointment Service
* **Healthcare.in/patients** for the Patient Service

**Project Structure:**



* **Appointment service:** has a Dockerfile which is used to build an image by copying index.js and package. Json
* **Patient-service:** has a Dockerfile which is used to build an image by copying index.js and package. Json
* **appointment-deployment.Yaml:** which defines a Kubernetes deployment configuration for the appointment service

****

* **Switch to name space:**

**kubectl config set-context --current --namespace=today-ns**

* **Apply configuration:**

**kubectl apply -f appointment-deployment. Yaml**

**apiVersion** and **kind**: The API version is apps/v1, which indicates it's using the newer version for deploying applications, and the kind is Deployment, meaning this is a deployment configuration.

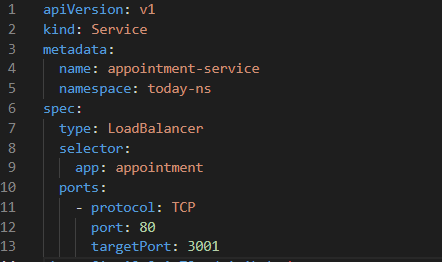
**metadata**: This section includes the name of the deployment (appointment-deployment) and the namespace (today-ns) where the deployment will be created. The namespace helps organize resources in Kubernetes.

**spec**: This defines the specifications of the deployment, such as:

* **replicas**: The deployment specifies 1 replica of the appointment service pod.
* **selector**: A selector matches the label **app: appointment**, ensuring that the pod created by this deployment has the appropriate label.
* **template**: This is the pod template. It describes the pod that will be created by the deployment.
  + **metadata**: Defines the label for the pod (**app: appointment**).
* **spec**: The specification for the pod's containers and other settings.
* **initContainers**: This section defines an init container, which runs before the main container starts. The init container uses the busybox image and executes a simple command (echo init container initializing && sleep 5), simulating an initialization step.
* **containers**: The main container uses the balure/appointment image and listens on port 3001. The container's resource requests and limits are defined for memory and CPU to ensure it has enough resources but doesn't exceed usage limits.
  + **requests**: Specifies the minimum resources required by the container.
  + **limits**: Defines the maximum resources that the container can use.
* **liveness Probe**: A liveness probe checks if the container is running properly. It sends an HTTP request to the /health endpoint on port 3001. If the probe fails, Kubernetes will restart the container.
* **readiness Probe**: A readiness probe checks if the container is ready to accept traffic. It also uses the /health endpoint, but it has different timings, with a shorter delay to allow the service to be considered ready after initialization.
* Just like the **appointment-deployment. yaml**, we have a **patient-deployment. yaml** that defines the deployment for the patient service with similar configurations for container setup, resources, and health probes.
* **Apply configuration:**

**kubectl apply -f patient-deployment. Yaml**

* **appointment-service. Yaml:** defines a Kubernetes Service for the AppointmentService.

****

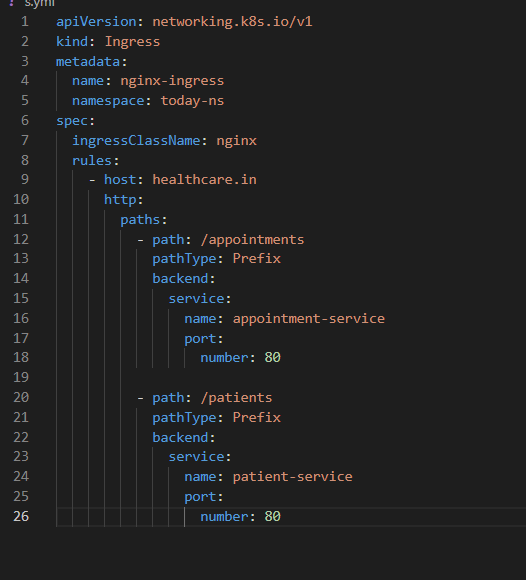
* **Apply configuration:**

**kubectl apply -f appointment-service. Yaml**

* **apiVersion** and **kind**: The API version is v1, which indicates it’s using the core Kubernetes API for services. The kind is Service, meaning this file is configuring a service that exposes the **appointment service**.
* **metadata**: This section includes the name of the service (appointment-service) and the namespace (today-ns) where it resides. The namespace helps in organizing and isolating resources in Kubernetes.
* **spec**: This section defines the configuration of the service:
  + **type**: The service is of type Load Balancer, meaning Kubernetes will provision an external load balancer (if available) that directs traffic to this service. This is useful for making the **appointment service** accessible from outside the Kubernetes cluster.
  + **selector**: The selector is used to identify which pods the service should route traffic to. Here, it matches pods with the label app: appointment, ensuring that traffic is sent to the **appointment service** pods.
  + **ports**:
    - The service listens on port 80 (the external port that clients will connect to).
    - Traffic is forwarded to port 3001 on the selected pods, which is where the **appointment service** is running.
    - The protocol used is TCP, which is typical for services exposing HTTP-based apps.
* Just like the **appointment-service. Yaml**, we have a **patient-service. Yaml** that defines the service for the patient service with similar configurations for port setup, selector, and service type.
* **Apply configuration:**

**kubectl apply -f appointment-deployment. Yaml**

**nginx-ingress. Yaml**: defines a Kubernetes Ingress resource that handles routing for the Appointment Service and Patient Service using path-based routing.

****

* **Apply configuration:**

**kubectl apply -f nginx-ingress. yaml**

**apiVersion** and **kind**: The API version is **networking.k8s.io/v1**, which is the latest API version for networking resources like Ingress. The kind is Ingress, which defines rules for routing HTTP(S) traffic to services within the Kubernetes cluster.

**metadata:** This section defines the name of the Ingress resource (nginx-ingress) and the namespace (today-ns) where the resource is located.

**spec**: This section contains the rules and configurations for the Ingress:

* **ingressClassName**: Specifies the Ingress controller to use. Here, it is set to **nginx**, meaning the traffic will be routed using the **Nginx Ingress Controller**.
* **rules**: Defines the routing rules for incoming traffic:
  + **host**: Traffic directed to healthcare.in will be handled by this Ingress.
  + **http**: Under the HTTP configuration, two paths are defined:
* **/appointments**: Traffic to /appointments will be routed to the **appointment-service** on port 80. This is handled by the backend service.
* **/patients**: Traffic to /patients will be routed to the **patient-service** on port 80 This is handled by the backend service.
* **path Type**: The Prefix path type means that any URL starting with **/appointments** or **/patients** will match, and the traffic will be forwarded accordingly.

**All used commands:**

**# Docker Commands**

**# Build Docker Image**

**docker build -t <image-name> .**

**# Tag Docker Image**

**docker tag <image-name> <repository>:<tag>**

**# Push Docker Image to Docker Hub**

**docker push <repository>:<tag>**

**# Minikube Commands**

**# Install Minikube (Linux example)**

**curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64**

**sudo install minikube-linux-amd64 /usr/local/bin/minikube**

**# Start Minikube**

**minikube start**

**# Enable Minikube Addons (Ingress)**

**minikube addons enable ingress**

**# Create Namespace**

**kubectl create namespace today-ns**

**# Kubernetes Commands**

**# Apply Appointment Deployment**

**kubectl apply -f appointment-deployment. yaml**

**# Apply Patient Deployment**

**kubectl apply -f patient-deployment. yaml**

**# Apply Appointment Service**

**kubectl apply -f appointment-service. Yaml**

**# Apply Patient Service**

**kubectl apply -f patient-service. Yaml**

**# Apply Ingress**

**kubectl apply -f nginx-ingress. yaml**

**# Check Pods in Namespace**

**kubectl get pods -n today-ns**

**# Check Services in Namespace**

**kubectl get svc -n today-ns**

**# Check Ingress**

**kubectl get ingress -n today-ns**

**# Access Pods Logs (if needed)**

**kubectl logs <pod-name> -n today-ns**

**# Access Service Using curl (replace <port> with the actual port)**

**curl http://$(minikube ip):<port>/appointments**

**curl http://$(minikube ip):<port>/patients**

**# Edit /etc/hosts to Map Custom Hostname**

**sudo nano /etc/hosts**

**# Add the following line to the file:**

**# <minikube-ip> healthcare.in**

**# Access the Application Using Custom Host (after editing /etc/hosts)**

**curl http://healthcare.in/appointments**

**curl http://healthcare.in/patients**

**------------------------------------------------------------**

**# Check all namespaces**

**kubectl get namespaces**

**# Delete resources by file or by name**

**kubectl delete -f <resource-file>.yaml**

**kubectl delete deployment <deployment-name> -n today-ns**

**kubectl delete service <service-name> -n today-ns**

**kubectl delete ingress <ingress-name> -n today-ns**

**# Check logs of a specific pod**

**kubectl logs <pod-name> -n today-ns**

**# Access Minikube dashboard**

**minikube dashboard**

**# Access the shell of a running pod for debugging**

**kubectl exec -it <pod-name> -n today-ns -- /bin/bash**

**# Expose a service manually if LoadBalancer is not working in Minikube**

**kubectl expose deployment <deployment-name> --type=NodePort --name=<service-name> -n today-ns**

**# Check resource usage in Kubernetes**

**kubectl top pods -n today-ns**

**kubectl top nodes**

**# Check deployment status**

**kubectl rollout status deployment/<deployment-name> -n today-ns**

**-------------------------------------------------------**