**3.Prepare a Simple Web Application Kubernetes Resource Template:**

**1. Design a Kubernetes resource template for deploying a simple web application.**

**2. Ensure the application is accessible from the internet.**

**3. Incorporate security compliance measures in the design.**

👉 The goal of this assignment is to prepare a simple Kubernetes resource template that:

1. Deploys a simple web application using NGINX.
2. Makes the application accessible externally via a LoadBalancer service.
3. Incorporates security best practices like running containers as non-root users and setting resource limits.
4. Ensures application health and resilience using readiness and liveness probes.
5. Supports scalability by deploying multiple pod replicas.

👉 Real-World Scenario Examples

* **E-commerce Platform**: During high traffic events like Black Friday, Kubernetes scales web servers to meet customer demand without downtime.
* **Microservices Architecture:** Enterprises deploy dozens of microservices with strict security and resource policies, managed consistently with Kubernetes and Terraform.
* **Global SaaS Provider:** A SaaS company maintains high availability and fault tolerance across regions using Kubernetes’ self-healing features.

👉 **1. Design a Kubernetes resource template for deploying a simple web application:**

**ConfigMap: nginx\_index**

* Stores the HTML content served by the nginx web server.
* Key points:

Name: nginx-index

Contains index.html with custom welcome message.

Allows easy updating of the web page without rebuilding container images.

**Deployment: webapp**

* Manages the pods running the nginx container serving the webpage.
* Key design features:

Replicas: 3 pods for scalability and high availability.

Selectors: Pods labeled app=webapp to associate with this deployment.

Pod Template: Defines the containers and volumes.

Security Context: Runs container as non-root user (UID 1000) for security compliance.

Container:

* + - Uses official nginx: latest image.
    - Listens on port 80.
    - Mounts the index.html file from ConfigMap as a read-only volume at nginx's default web directory.

Probes:

* + - Readiness probe ensures traffic is only routed to ready containers.
    - Liveness probe ensures unhealthy containers are restarted automatically.

Resource requests and limits for controlled CPU and memory usage.

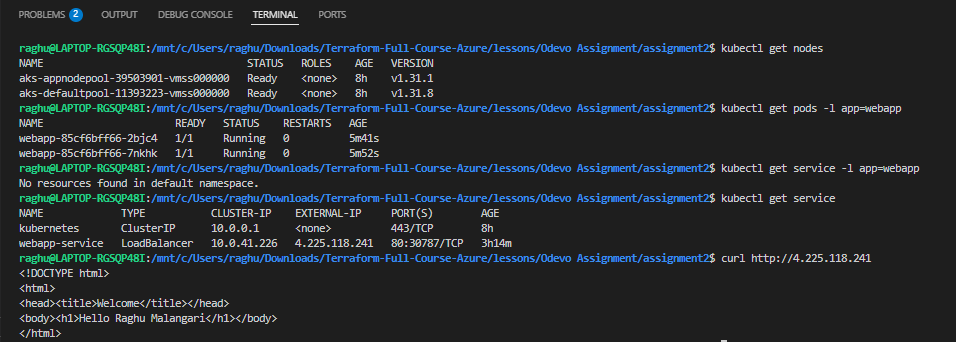
👉 **2. Ensure the application is accessible from the internet**

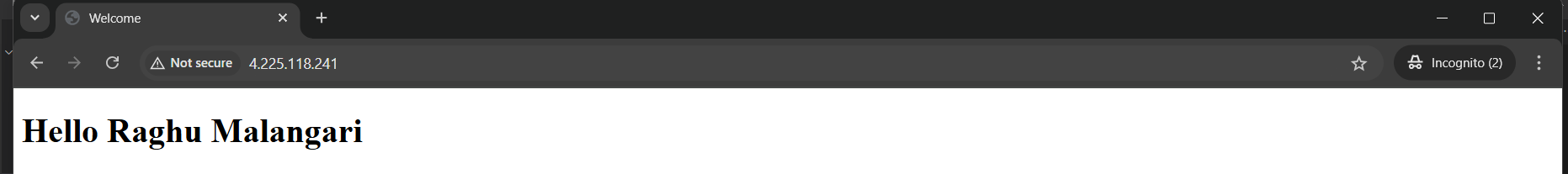
**Service: webapp\_service**

* Exposes the deployment pods to the internet.
* Type: LoadBalancer to provision cloud provider external IP.
* Selects pods with label app=webapp.
* Forwards external traffic on port 80 to pod container port 80.

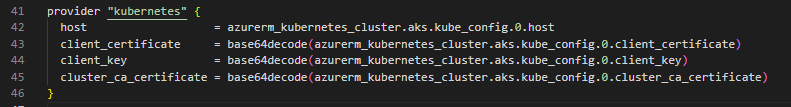
👉 **3. Incorporate security compliance measures in the design**

* Run containers as non-root user to follow best security practices.
* Mounted ConfigMap volume is read-only to prevent modifications at runtime.
* Liveness and readiness probes ensure the app is healthy and ready before receiving traffic.
* Resource limits prevent a container from consuming excessive resources, protecting cluster stability**.**

****

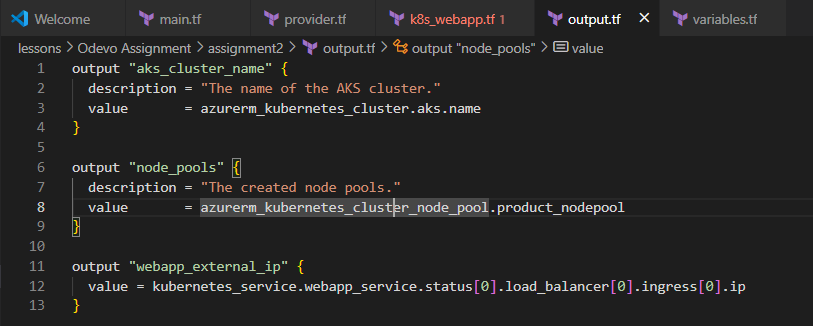
****

**🔌provider.tf**

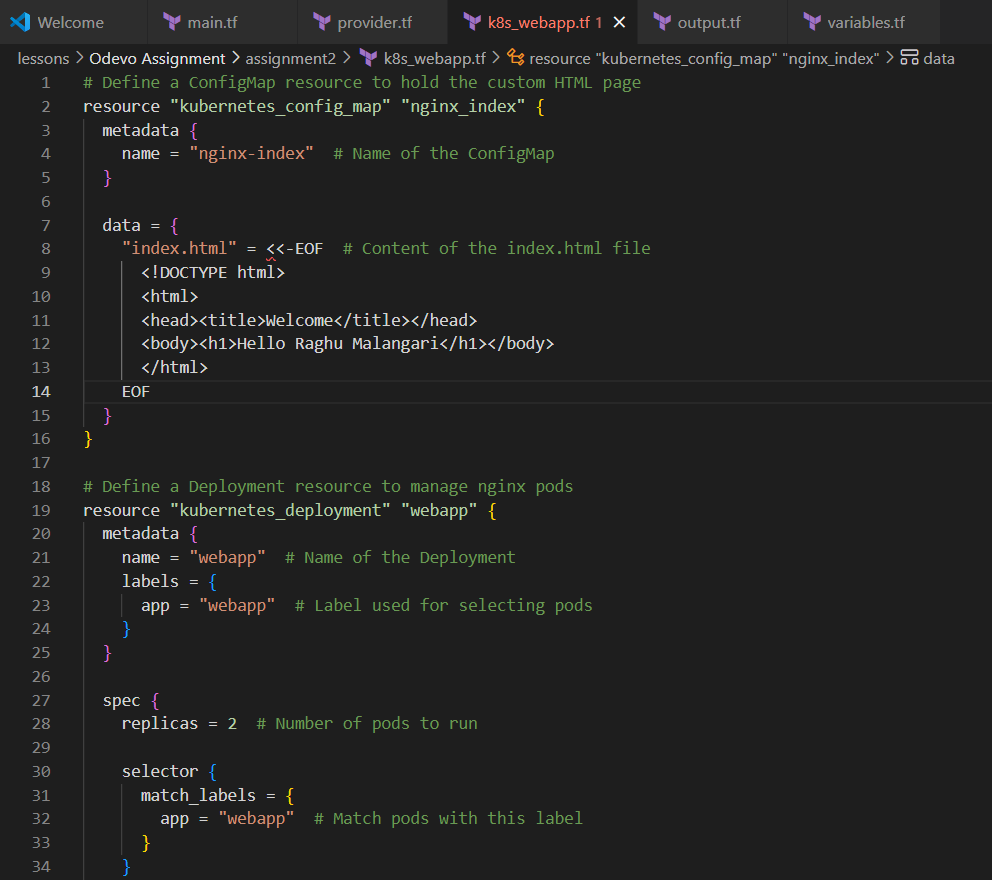
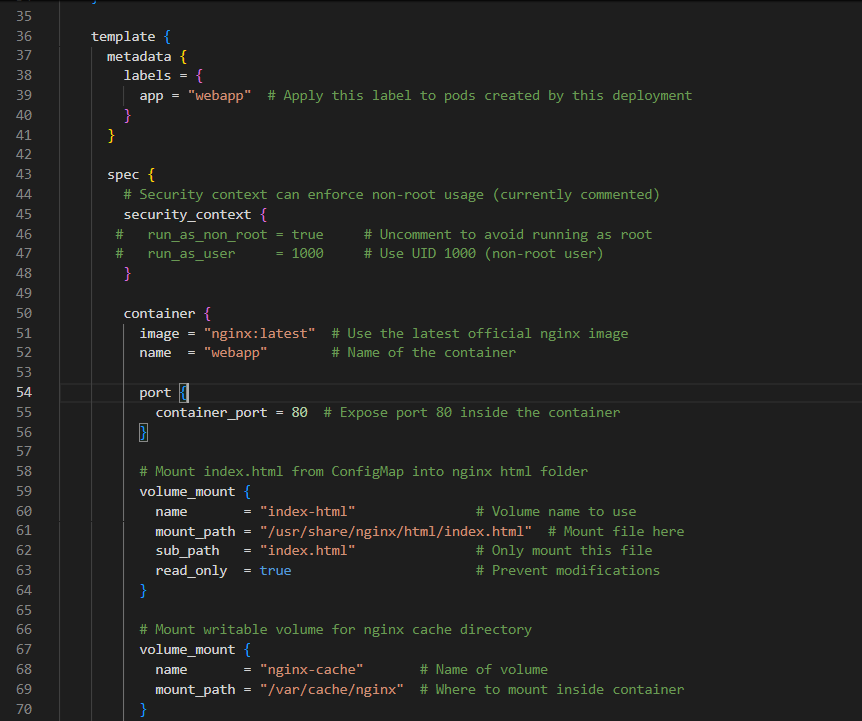
****

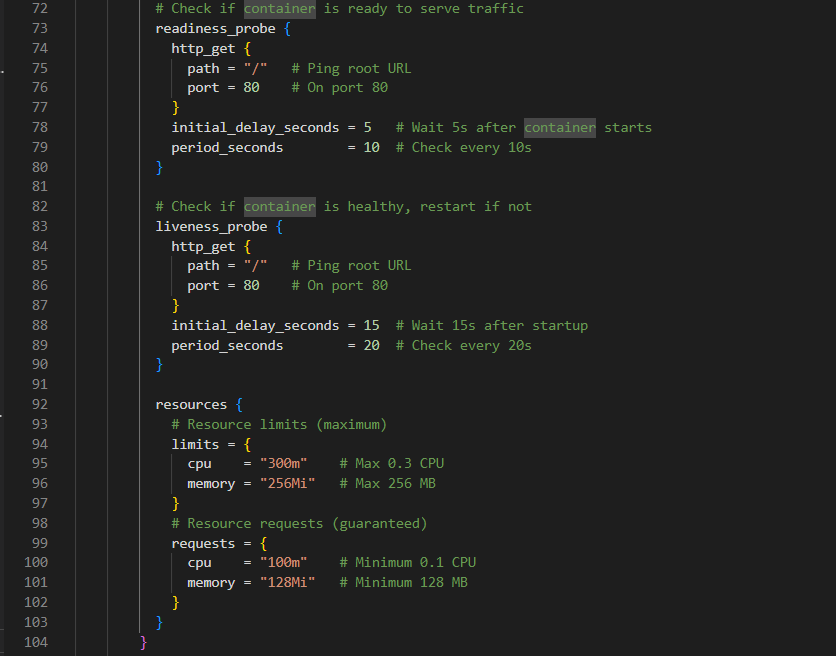
* Connects the Kubernetes provider to your AKS cluster using azurerm\_kubernetes\_cluster.aks.
* Uses client certificate authentication.
* Decodes the certificates which are base64 encoded by Azure**.**

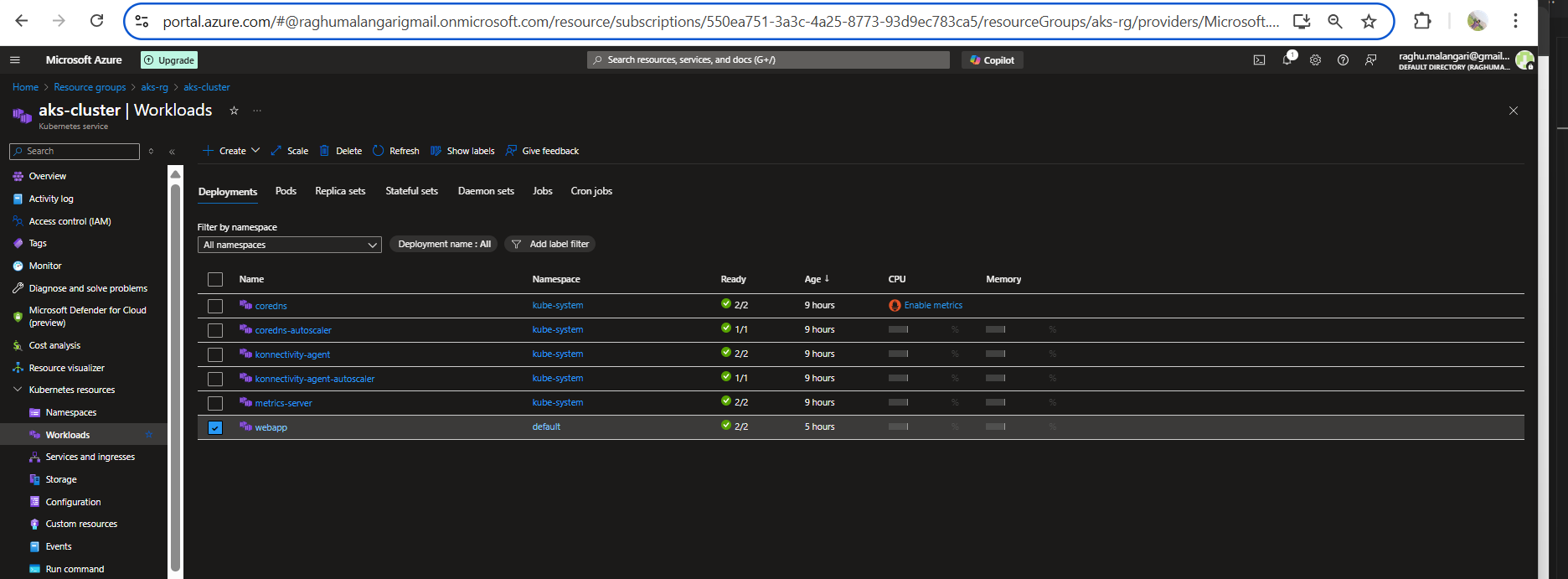
**🎯Output.tf**

****

**🏗️K8s\_webapp.tf**

**** ****

****

****

**Resources:**  
<https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/kubernetes_cluster>

https://learn.microsoft.com/en-us/cli/azure/?view=azure-cli-latest