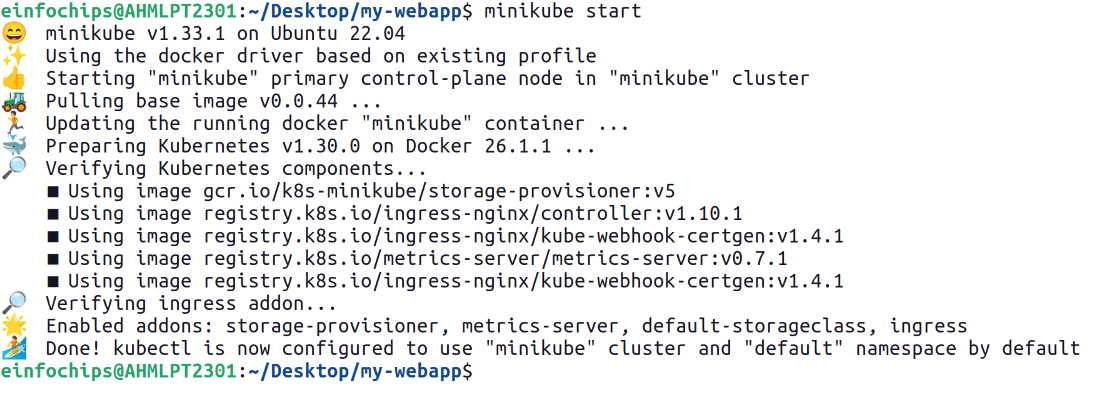
#### **Project Overview**

Participants are required to deploy a simple static web application on a Kubernetes cluster using Minikube, set up advanced ingress networking with URL rewriting and sticky sessions, and configure horizontal pod autoscaling to manage traffic efficiently. The project will be divided into stages, with each stage focusing on specific aspects of Kubernetes ingress, URL rewriting, sticky sessions, and autoscaling.

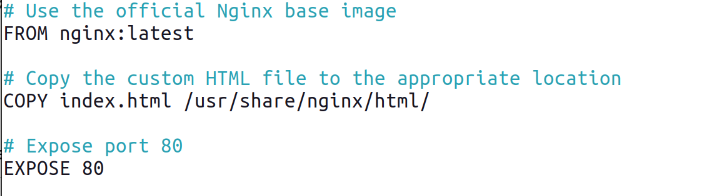
#### **Requirements and Deliverables**

### **Stage 1: Setting Up the Kubernetes Cluster and Static Web App**

1. **Set Up Minikube:**



1. **Deploy Static Web App:**
   * Create a Dockerfile for a simple static web application.



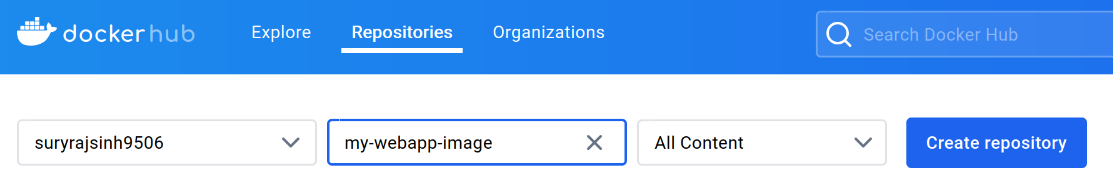
Here we use nginx as a base image to deploy our web app on it.

Below is the simple index.html :



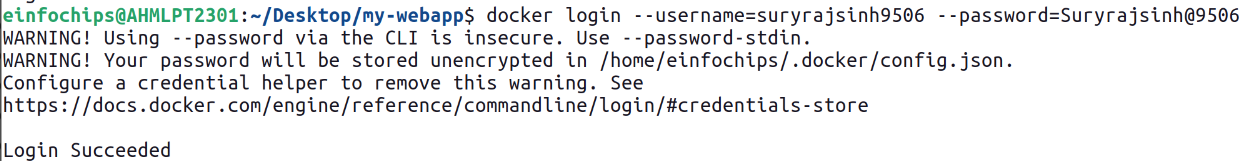
* + Build a Docker image for the static web application.

Here we want to push our image to the docker hub so that’s why we first have to login in our docker hub account and need to create one repository.

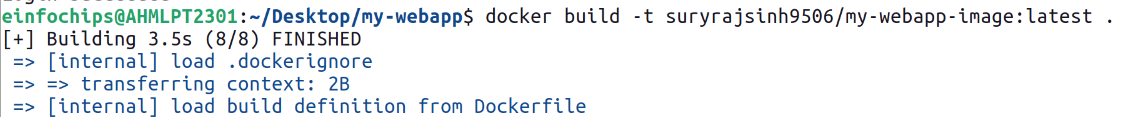


Here we create repository named my-webapp-image.

Now we have to login into the docker hub through terminal which is shown in below image.



Now while building docker image we give same name as our repository.



* + Push the Docker image to Docker Hub or a local registry.

1. **Kubernetes Deployment:**
   * Write a Kubernetes deployment manifest to deploy the static web application.
   * Write a Kubernetes service manifest to expose the static web application within the cluster.
   * Apply the deployment and service manifests to the Kubernetes cluster.

**Deliverables:**

* Dockerfile for the static web app
* Docker image URL
* Kubernetes deployment and service YAML files

### **Stage 2: Configuring Ingress Networking**

1. **Install and Configure Ingress Controller:**
   * Install an ingress controller (e.g., Nginx Ingress Controller) in the Minikube cluster.
   * Verify the ingress controller is running and accessible.
2. **Create Ingress Resource:**
   * Write an ingress resource manifest to route external traffic to the static web application.
   * Configure advanced ingress rules for path-based routing and host-based routing (use at least two different hostnames and paths).
   * Implement TLS termination for secure connections.
   * Configure URL rewriting in the ingress resource to modify incoming URLs before they reach the backend services.
   * Enable sticky sessions to ensure that requests from the same client are directed to the same backend pod.

**Deliverables:**

* Ingress controller installation commands/scripts
* Ingress resource YAML file with advanced routing, TLS configuration, URL rewriting, and sticky sessions

### **Stage 3: Implementing Horizontal Pod Autoscaling**

1. **Configure Horizontal Pod Autoscaler:**
   * Write a horizontal pod autoscaler (HPA) manifest to automatically scale the static web application pods based on CPU utilization.
   * Set thresholds for minimum and maximum pod replicas.
2. **Stress Testing:**
   * Perform stress testing to simulate traffic and validate the HPA configuration.
   * Monitor the scaling behavior and ensure the application scales up and down based on the load.

**Deliverables:**

* Horizontal pod autoscaler YAML file
* Documentation or screenshots of the stress testing process and scaling behavior

### **Stage 4: Final Validation and Cleanup**

1. **Final Validation:**
   * Validate the ingress networking, URL rewriting, and sticky sessions configurations by accessing the web application through different hostnames and paths.
   * Verify the application's availability and performance during different load conditions.
2. **Cleanup:**
   * Provide commands or scripts to clean up the Kubernetes resources created during the project (deployments, services, ingress, HPA).

**Deliverables:**

* Final validation report documenting the testing process and results
* Cleanup commands/scripts