### **Name: K K VISHAL**

### **Roll no : 22csr241**

### **Step-by-Step Guide to Deploying a Spring Boot Application with Docker and Kubernetes**

### 

### **1. Initialize and Clone the Repository**

Initializes a new Git repository.

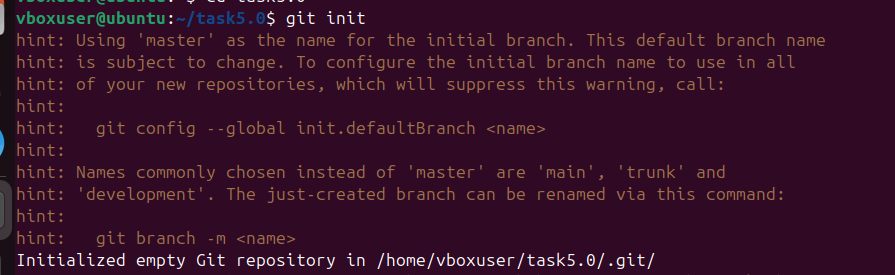
Clones the Spring Framework PetClinic project from GitHub.

**Code:**

git init

git clone "https://github.com/AranganathanPrakash/spring-framework-petclinic"

**Screenshot:**

****

**2. Navigate to the Project Directory**

Moves into the cloned repository folder.

**Code:**

cd spring-framework-petclinic

### 

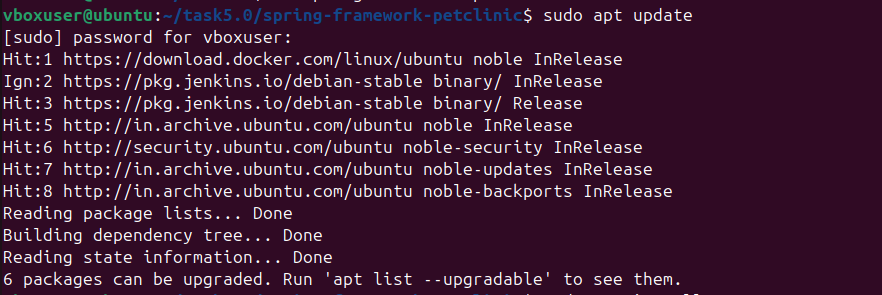
### **3. Update System Packages**

Updates the package list to ensure the latest versions are available.

**Code:**

sudo apt update

**Screenshot:**

****

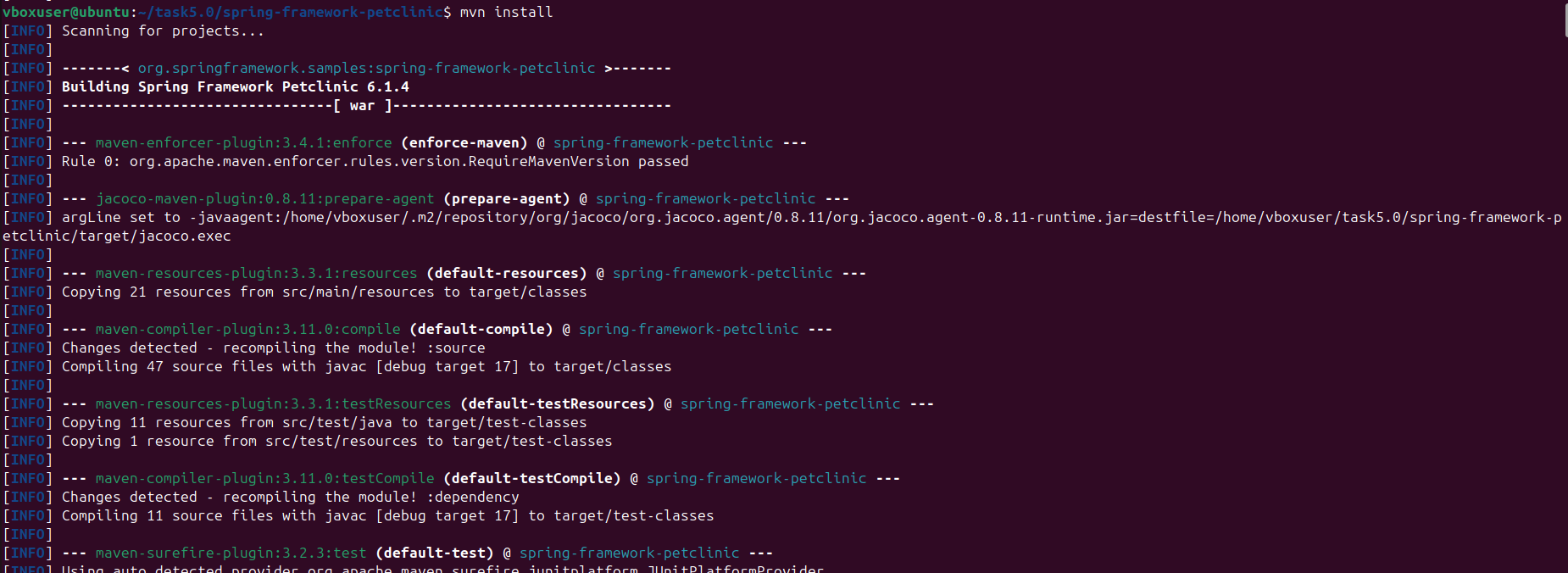
**4. Install Maven**

Installs Apache Maven, required for building the Spring Boot application.

**Code:**

sudo apt install maven

**Screenshot:**

****

### **5. Verify Maven Installation**

Checks if Maven is installed correctly and displays the version.

**Code:**

mvn --version

**ScreenShot:**

****

**6. Run Tests (Optional)**

Executes unit tests to ensure the application works correctly.

**Code:**

mvn test

**Screenshot:**

****

**7. Clean and Build the Application**

**mvn clean: Cleans previous builds.**

**mvn install: Compiles and packages the application.**

**mvn package: Generates the final JAR/WAR file in the target/ directory.**

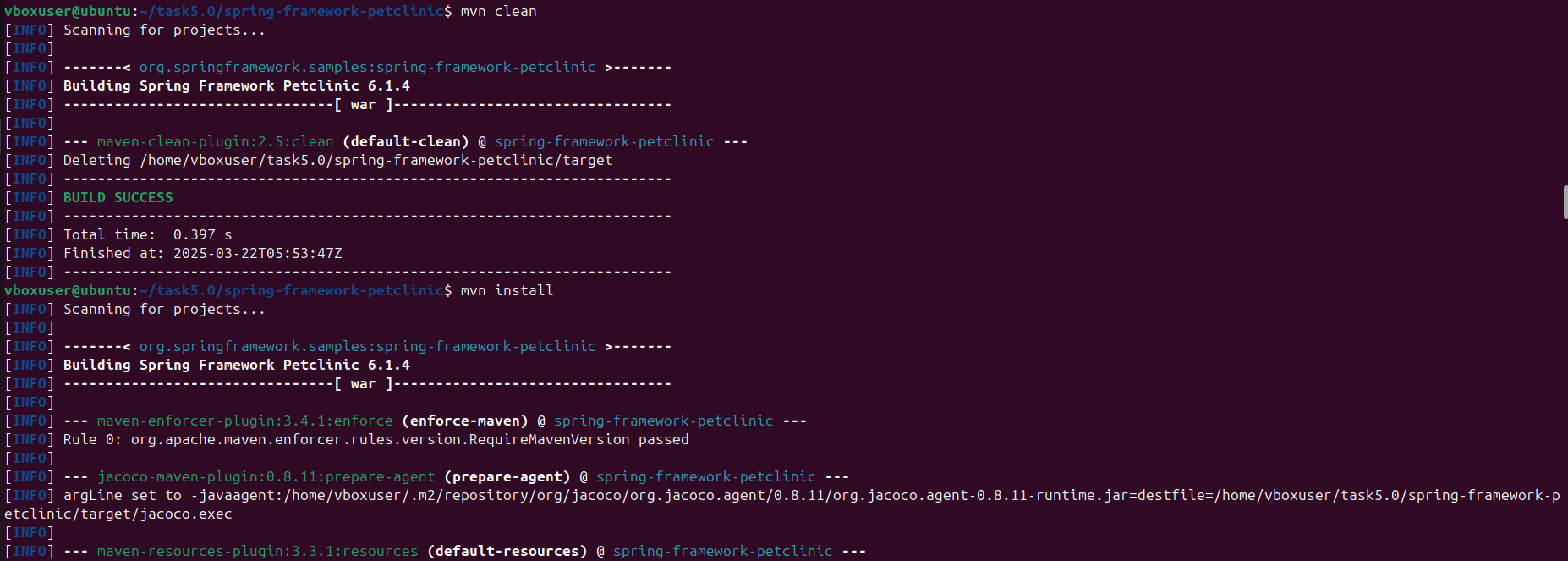
**Code:**

mvn clean

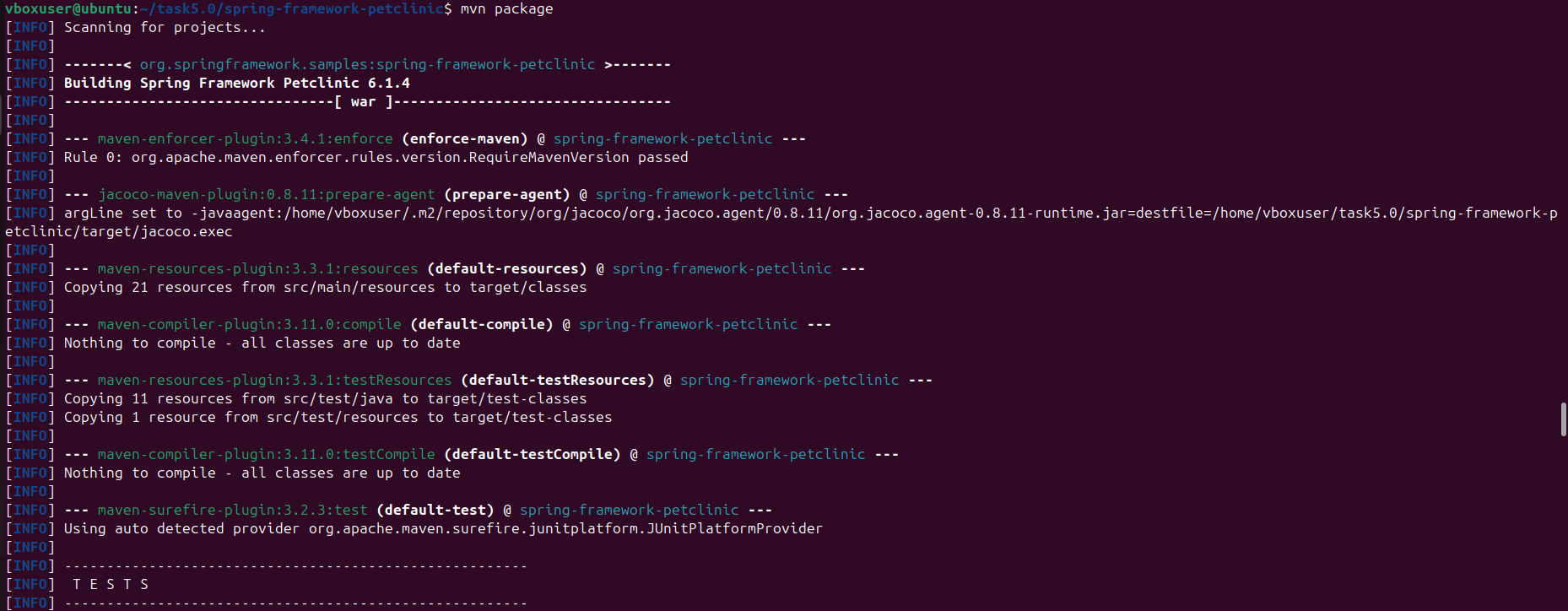
mvn install

mvn package

**Screenshot:**

****

****

****

**8. Verify the Built Application**

Navigates to the target folder where the compiled application is stored.

**Code:**

cd target

ls

cd ..

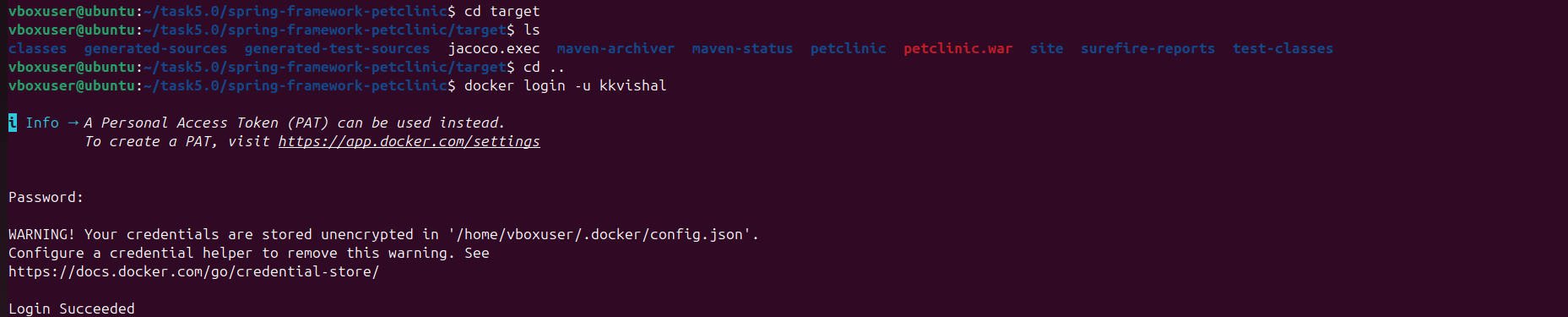
**9. Login to Docker**

Logs into Docker Hub to push container images.

**Code:**

docker login -u kkvishal

**Screenshot:**

****

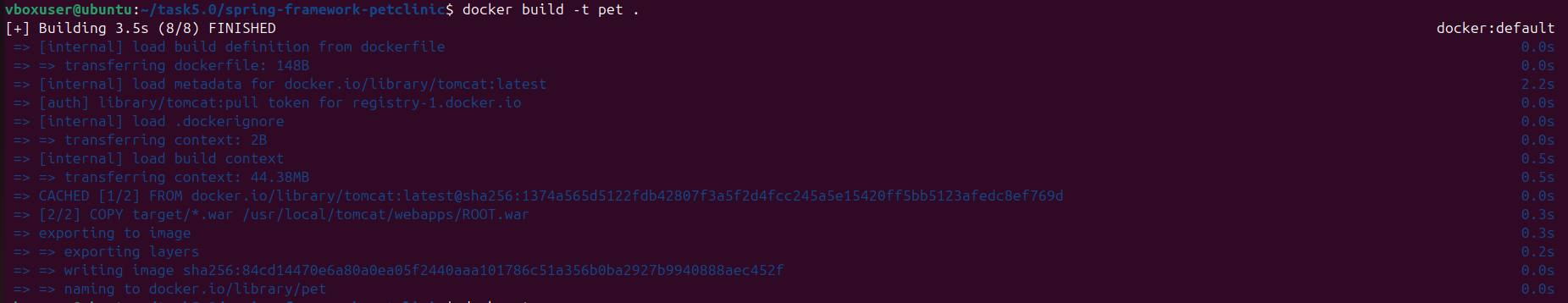
### **10. Build Docker Image**

### Builds a Docker image with the tag pet from the project directory.

**Code:**

docker build -t pet .

**Screenshot:**

****

### **11. Tag and Push Image to Docker Hub**

Tags the image for Docker Hub.

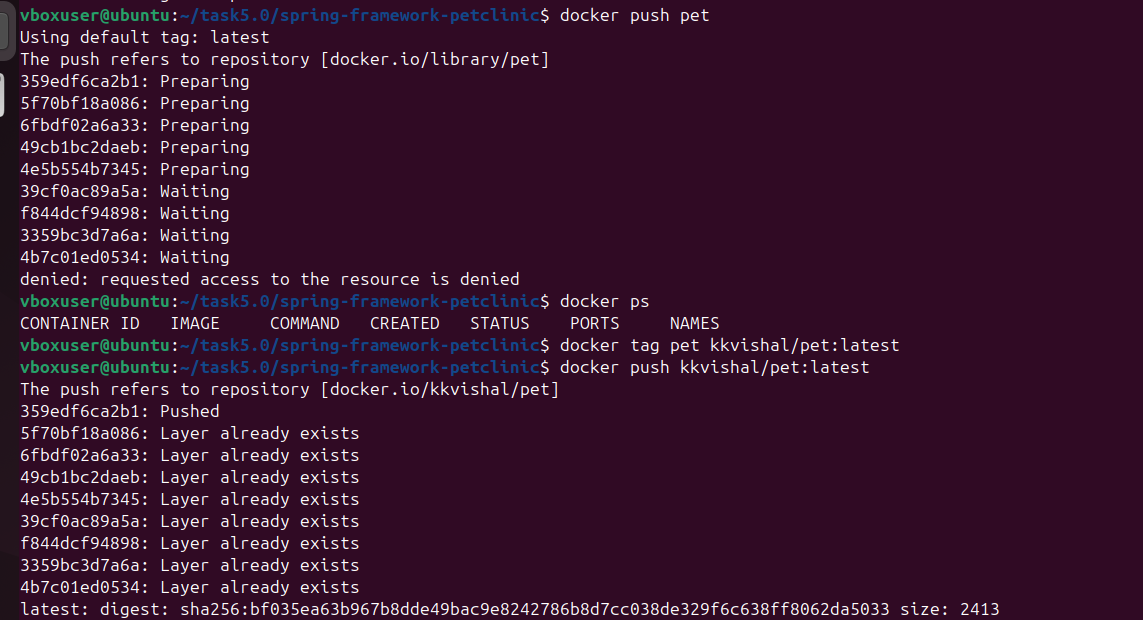
Pushes the image to your Docker Hub repository.

**Code:**

docker tag pet kkvishal/pet:latest

docker push kkvishal/pet:latest

**Screenshot:**

****

**12. Start Minikube**

Starts a Minikube cluster for Kubernetes.

Checks if Minikube is running properly.

**Code:**

minikube start

minikube status

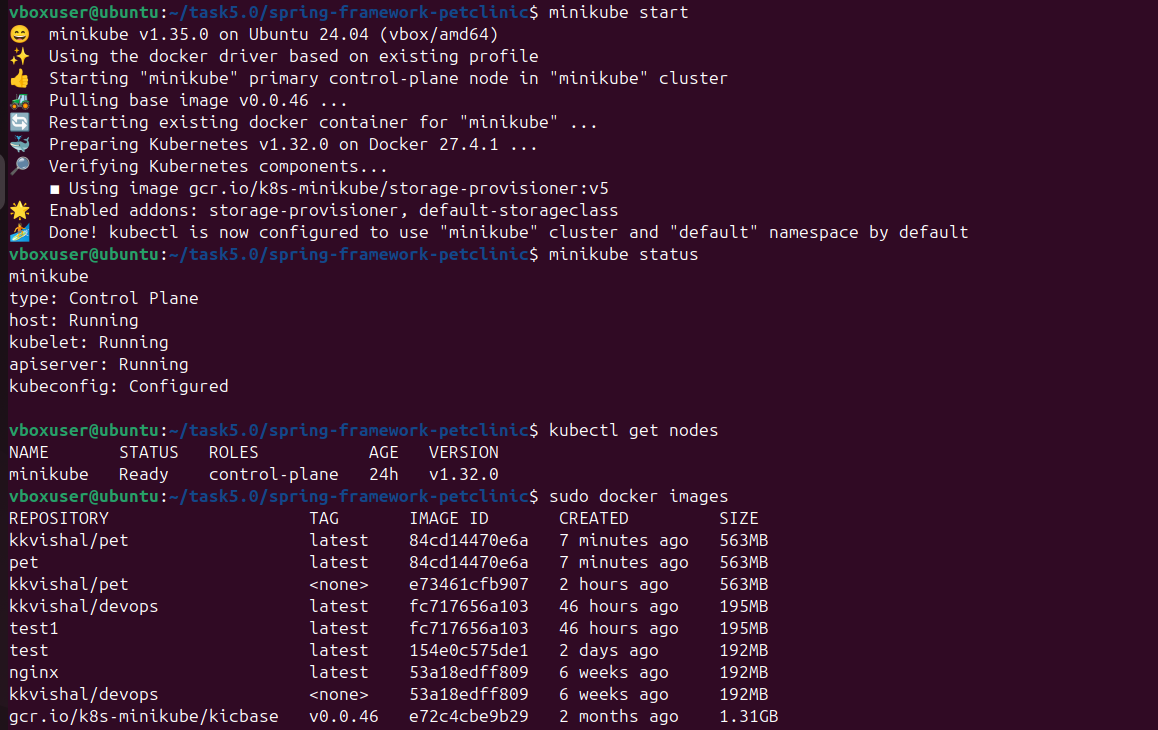
**13. Verify Kubernetes Nodes**

Lists available Kubernetes nodes.

**Code:**

kubectl get nodes

**ScreenShot:**

****

### **14. Deploy the Application on Kubernetes & Expose the Application**

Creates a Kubernetes deployment using your Docker image.

Exposes the deployment as a service, making it accessible via Minikube.

Lists all running pods to verify the deployment is successful.

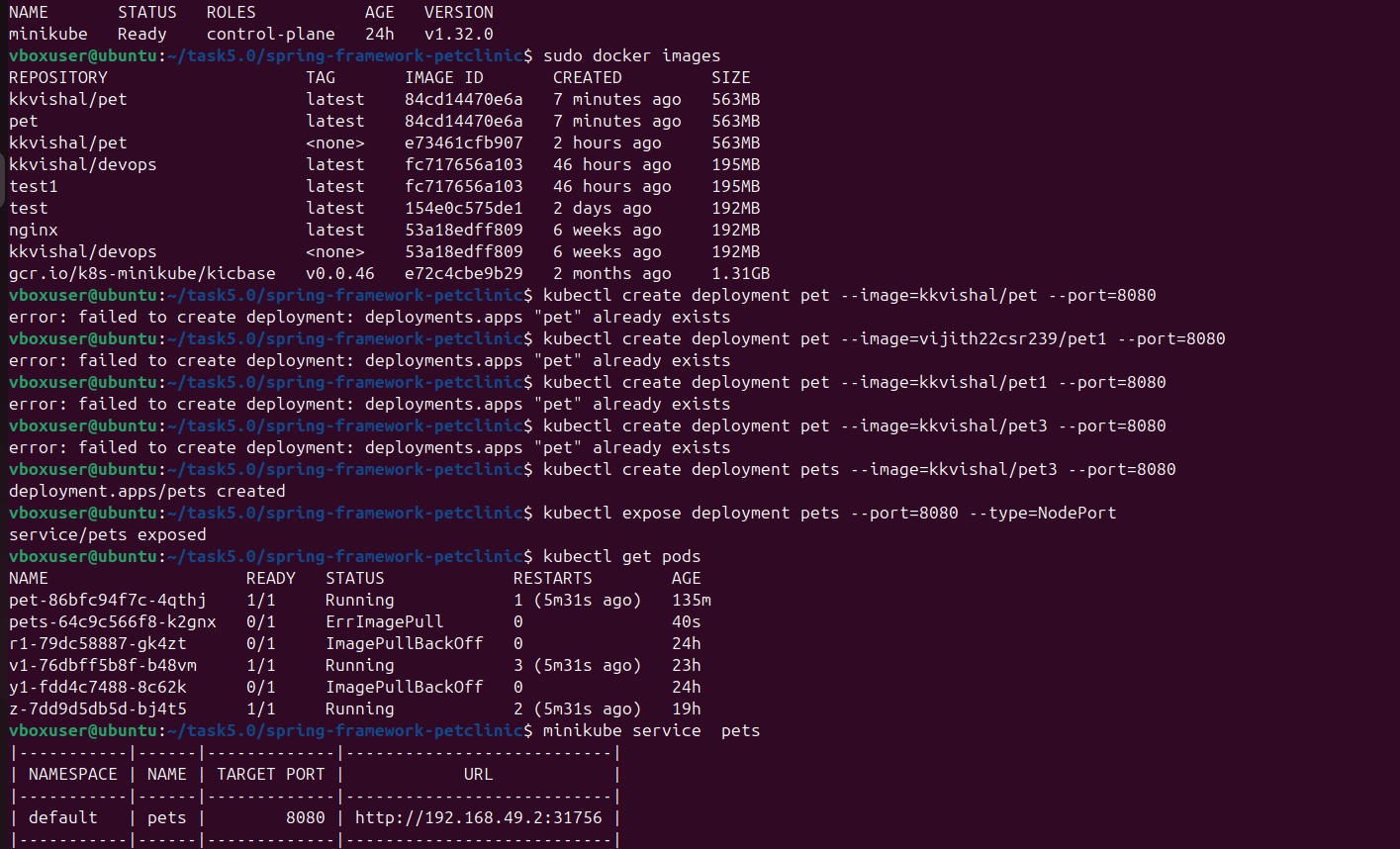
**Code:**

kubectl create deployment pet --image=kkvishal/pet --port=8080

kubectl expose deployment pet --port=8080 --type=NodePort

kubectl get pods

**Screenshot:**

****

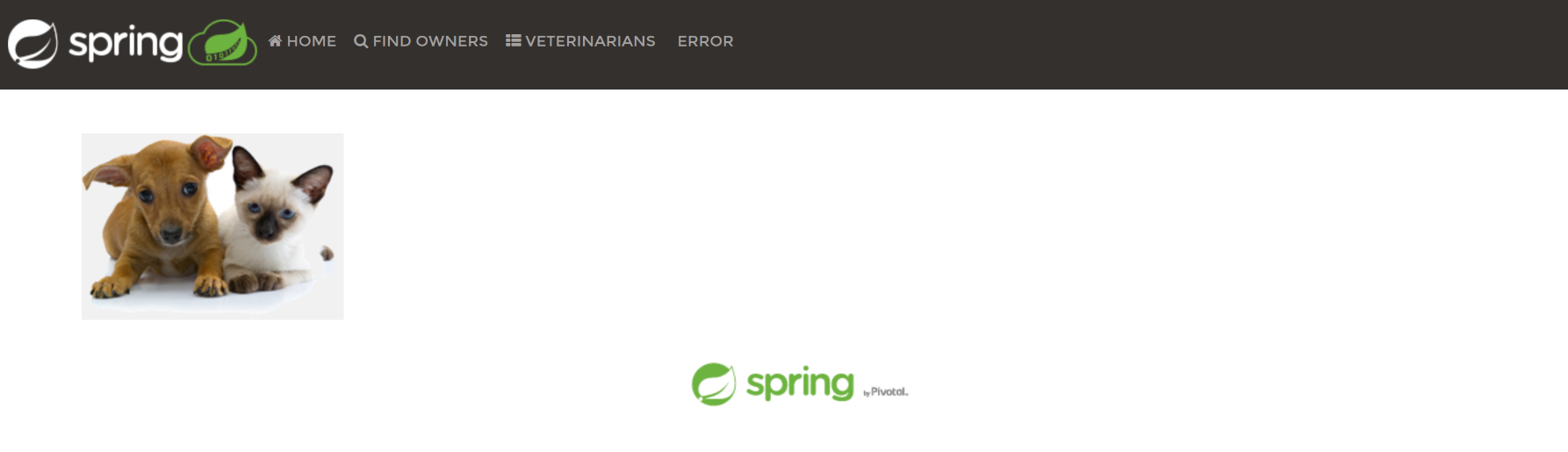
### **16. Access the Application**

Opens the application in the browser via Minikube.

**Code:**

minikube service pet

**Screenshot:**

****