**TASK-02 KUBERNETES**

**Date: 27/05/24**

**Q.1 Make a note on:**

* 1. **Pod**
  2. **Replica**
  3. **ReplicaSet**
  4. **Labels**
  5. **Namespace**

1. **Pod:**

* A pod is the smallest deployable unit in Kubernetes.
* It represents a single instance of a running process in your cluster.
* Pods can contain one or more containers that are tightly coupled and share resources, such as networking and storage.
* They are ephemeral by nature, meaning they can be created, destroyed, and replaced dynamically.

1. **Replicas:**

* Replicas refer to the number of identical copies of a pod that should be running at any given time.
* They are used to ensure high availability and scalability of applications.
* Replicas are typically defined in higher-level Kubernetes objects like ReplicaSets or Deployments.

1. **ReplicaSet:**

* A ReplicaSet ensures that a specified number of pod replicas are running at any given time.
* It acts as a higher-level abstraction over pods, managing their lifecycle and ensuring the desired number of replicas is maintained.
* ReplicaSets are generally used to achieve scaling and self-healing capabilities for stateless applications.

1. **Labels:**

* Labels are key-value pairs attached to Kubernetes objects such as pods, services, and deployments.
* They are used to organize and select subsets of objects based on user-defined criteria.
* Labels are highly flexible and can be used for various purposes including grouping, filtering, and identifying related resources within a cluster.

1. **Namespace:**

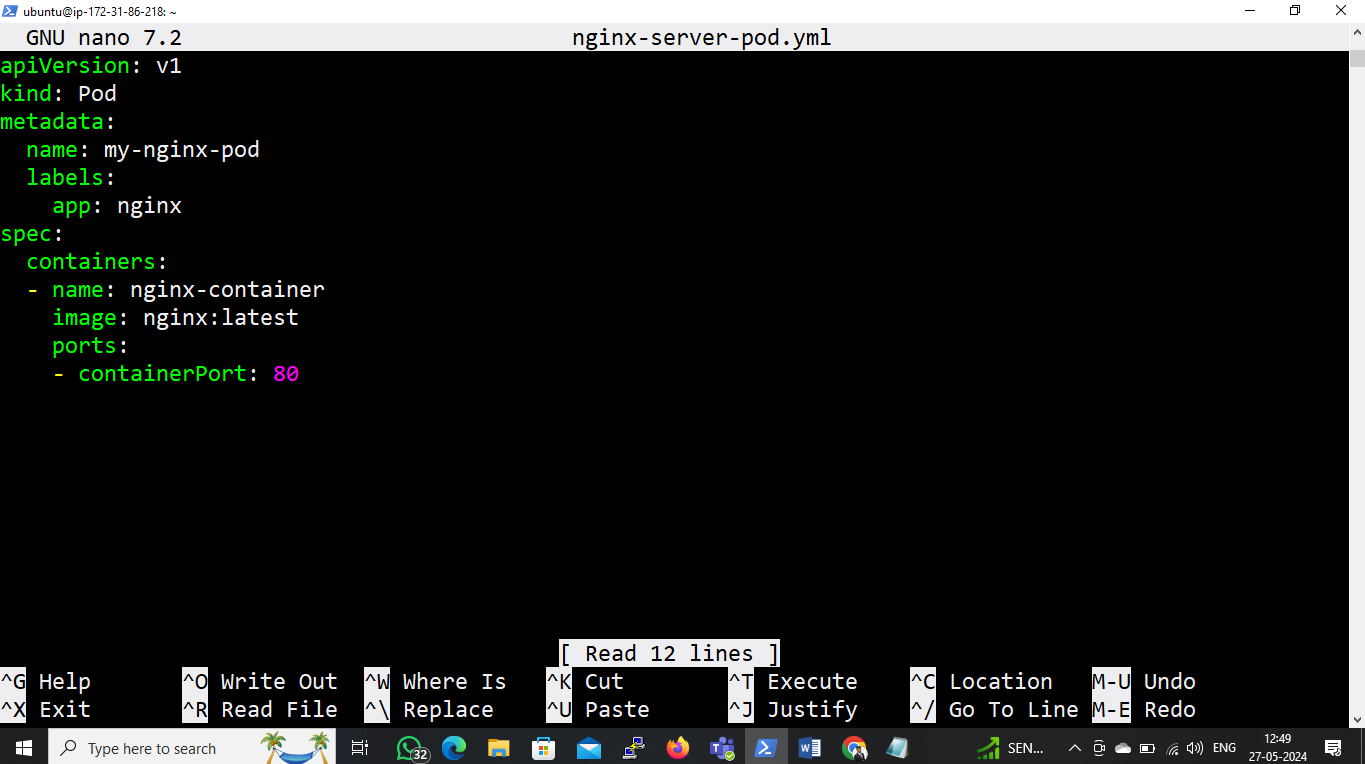
* Namespaces provide a way to logically divide cluster resources into virtual clusters within the same physical cluster.
* They are primarily used to create isolated environments for different teams, projects, or applications, allowing each to have its own scope of resources such as pods, services, and storage volumes.
* Namespaces also help in resource management, access control, and multi-tenancy scenarios in Kubernetes clusters.

**Q.2 Show in practical of RC uses with all types of health probes and their file.**

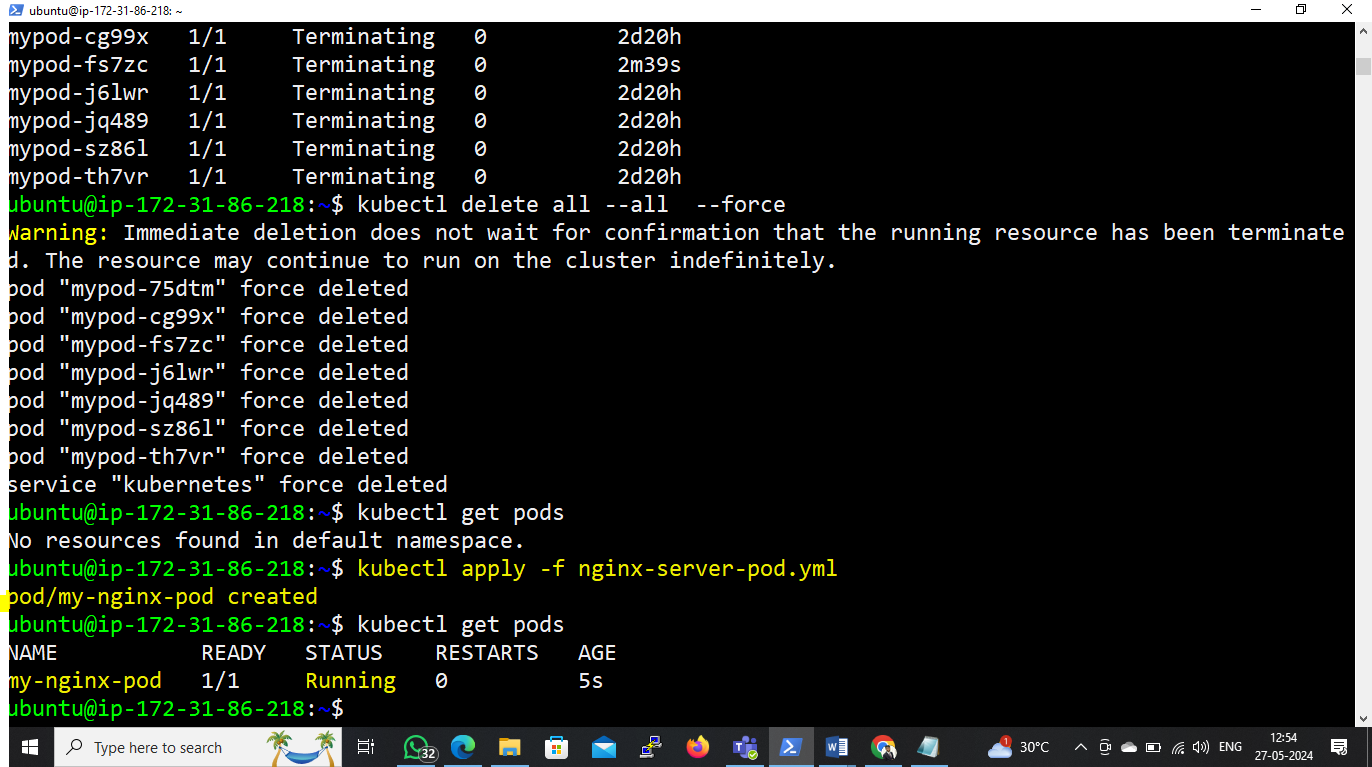
To demonstrate the use of ReplicaController (RC) with various types of health probes in Kubernetes.

Create a simple example with a Pod running a my-nginx-pod.

Below is the YAML manifest file for the Pod configuration:(nginx-server-pod.yml)



* kubectl apply -f nginx-server-pod.yml
* kubectl get pods



We'll use different types of health probes:

livenessProbe, readinessProbe, and startupProbe.

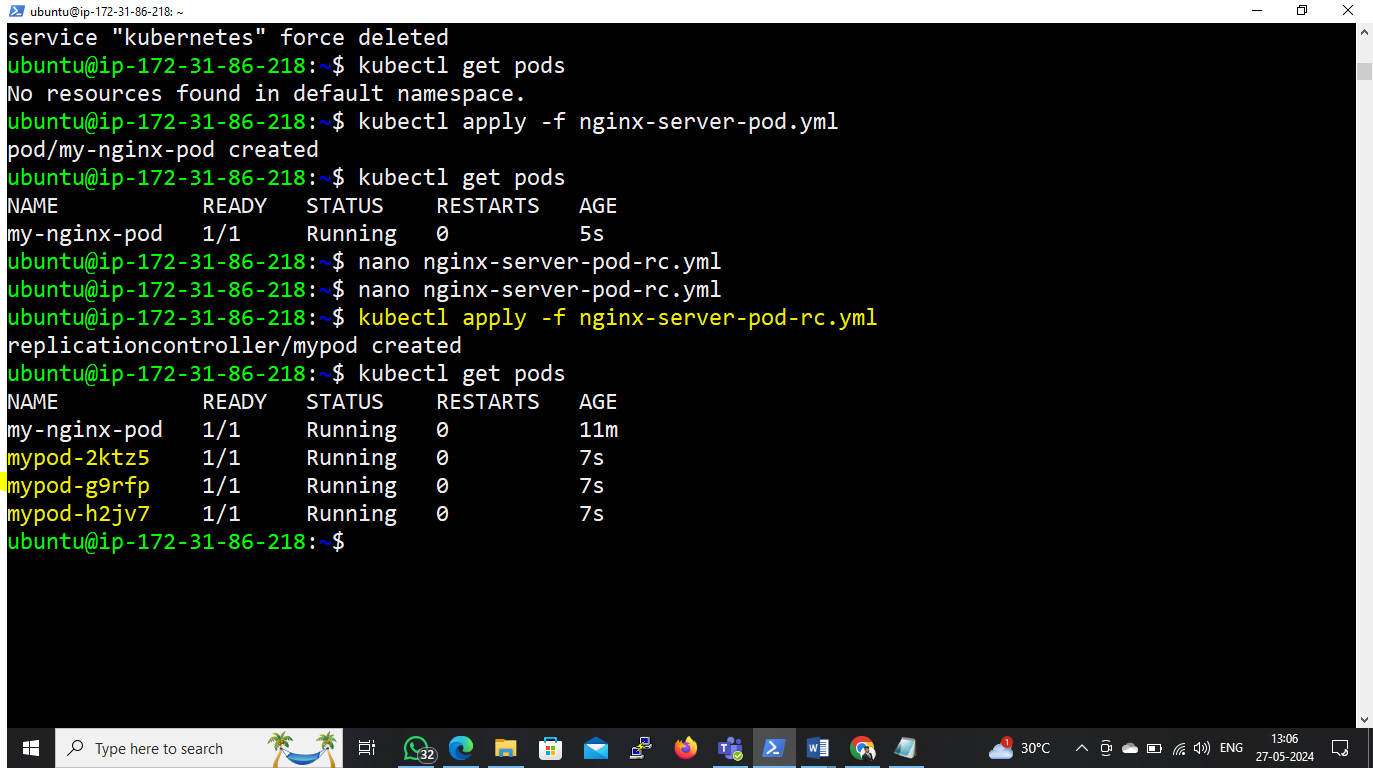
First create a ReplicaController file;

eg nginx-server-pod-rc.yml.



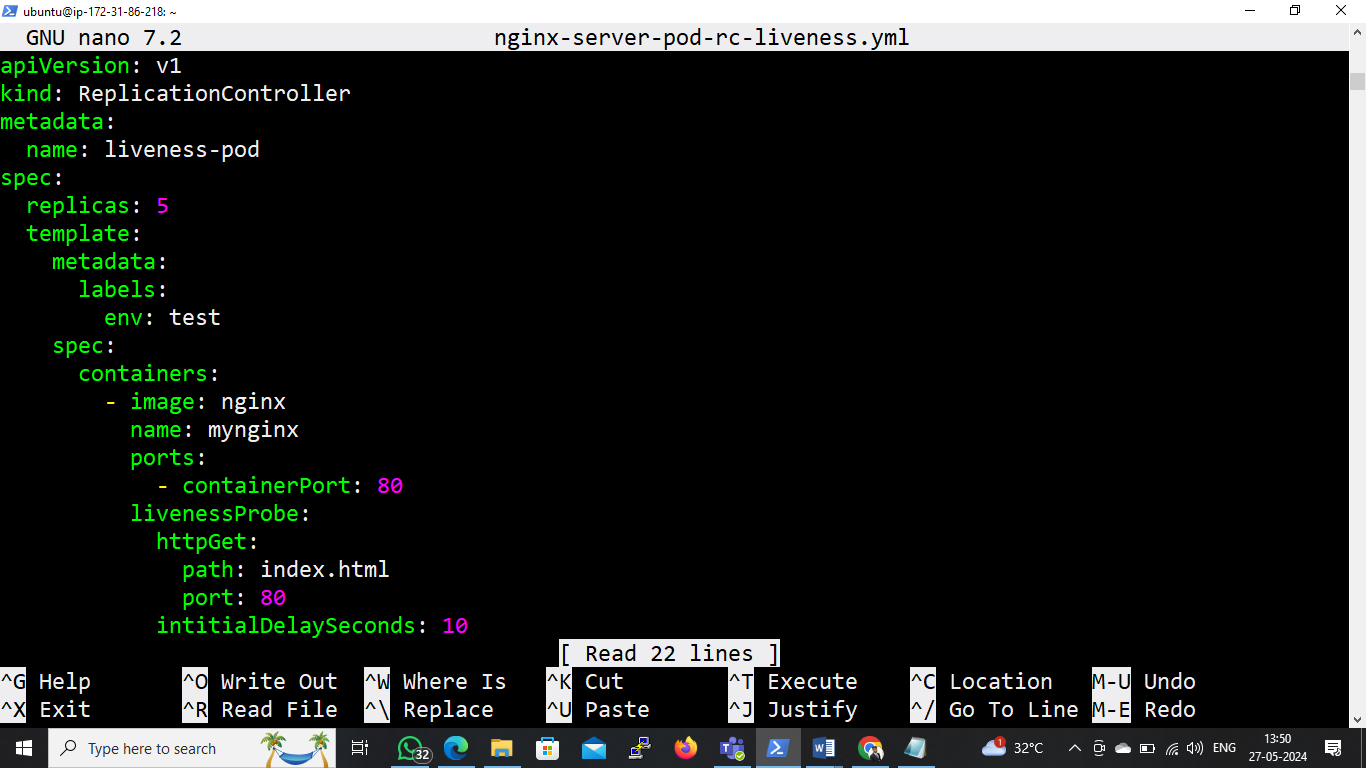
Apply this file;

* kubectl apply -f nginx-server-pod-rc.yml
* kubectl get pods

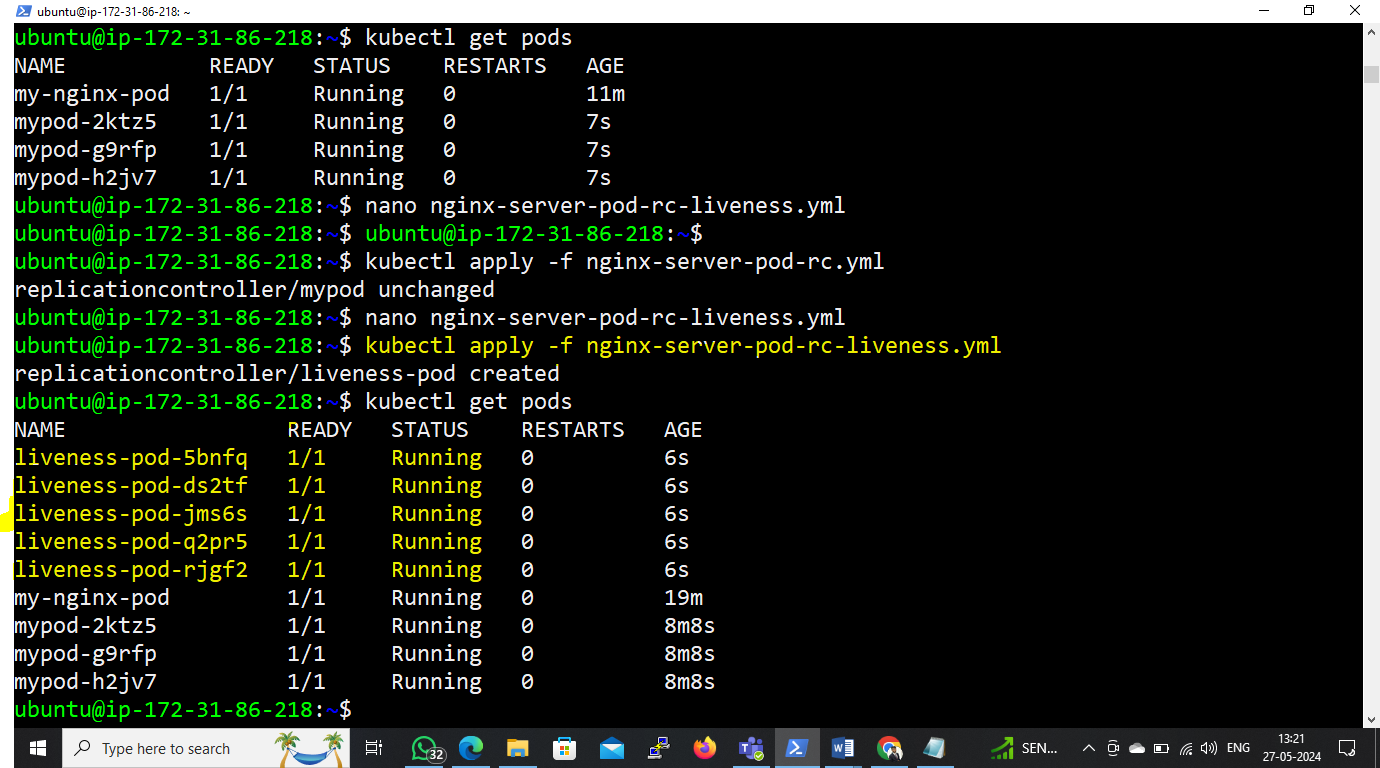


**Use of LivenessProbe:**

* **Liveness Probe:**
* **Purpose:** Determines whether the container in a Pod is running properly.
* **Functionality:** Periodically checks if the container is responsive and restarts it if it's not.
* **Typical Use Case:** Used to detect and recover from application-specific issues such as deadlocks or resource exhaustion that cause the container to become unresponsive.
* **Action on Failure:** If the liveness probe fails, Kubernetes restarts the container.
* Create a manifestfile of nginx-server-pod-rc-liveness.yml



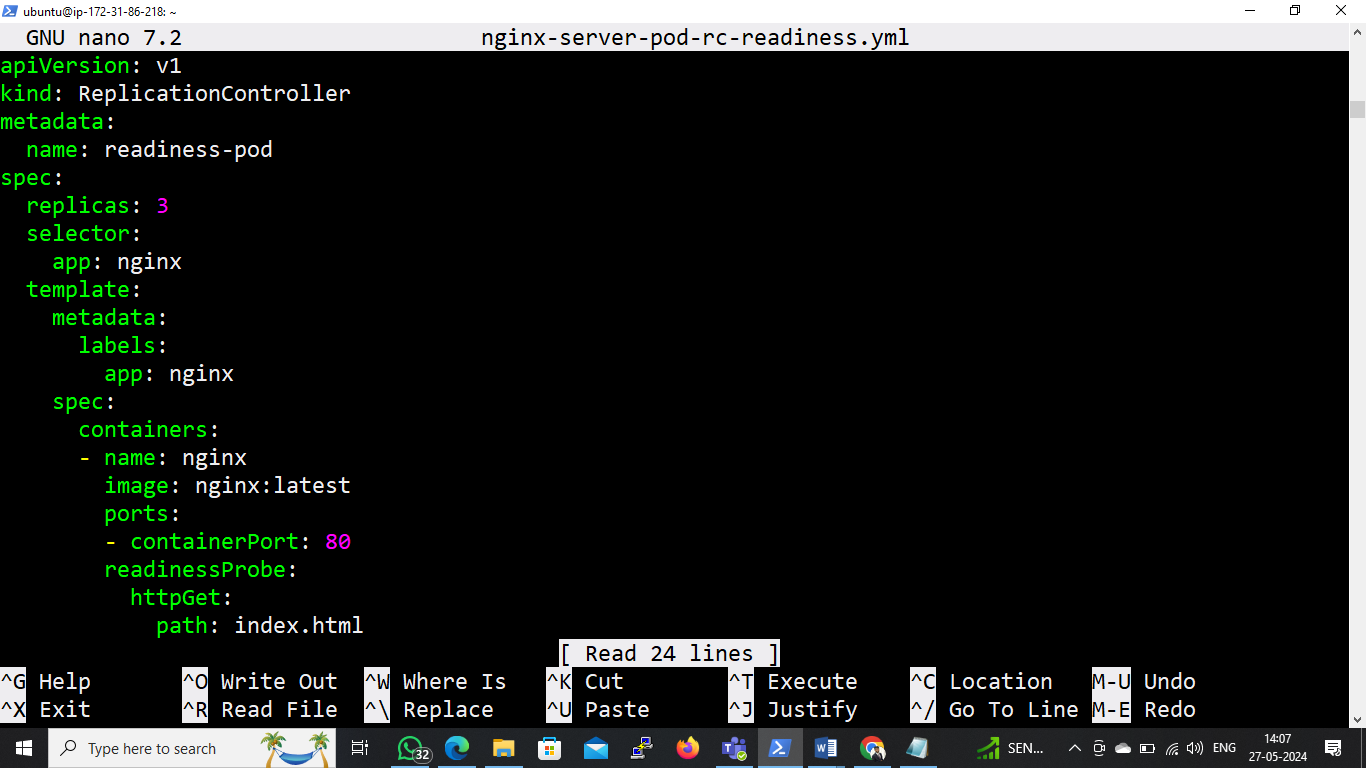
* kubectl apply -f nginx-server-pod-rc-liveness.yml
* kubectl get pods.



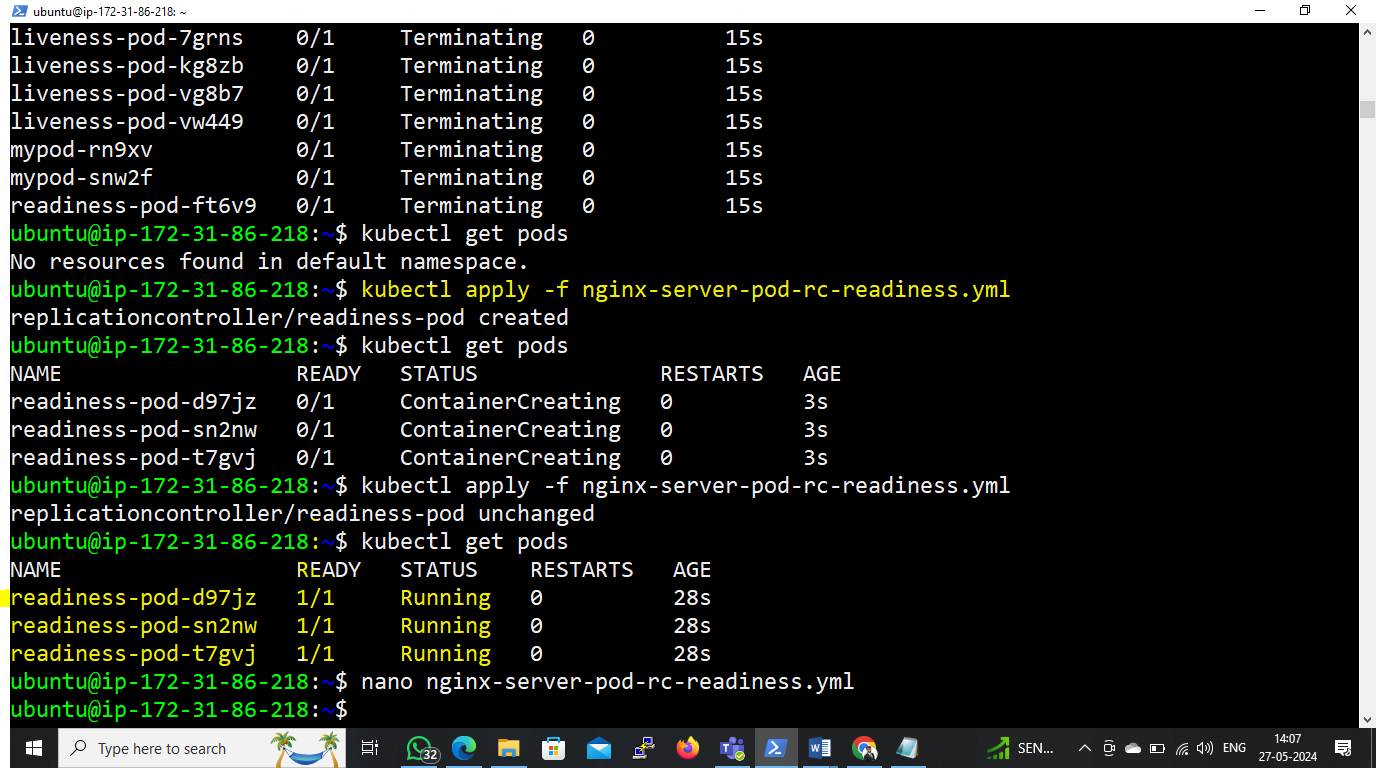
**Use of ReadinessProbe:**

* **Readiness Probe:**
* **Purpose:** Determines whether the container in a Pod is ready to serve traffic.
* **Functionality:** Periodically checks if the container is ready to receive requests and tells Kubernetes whether the Pod should receive traffic.
* **Typical Use Case:** Used to ensure that only healthy Pods receive traffic from services or load balancers. It helps avoid sending requests to Pods that are still initializing or experiencing issues.
* **Action on Failure:** If the readiness probe fails, the Pod is removed from service endpoints until it becomes ready again.

**#** Create a manifestfile **of** nginx-server-pod-rc-readiness.yml



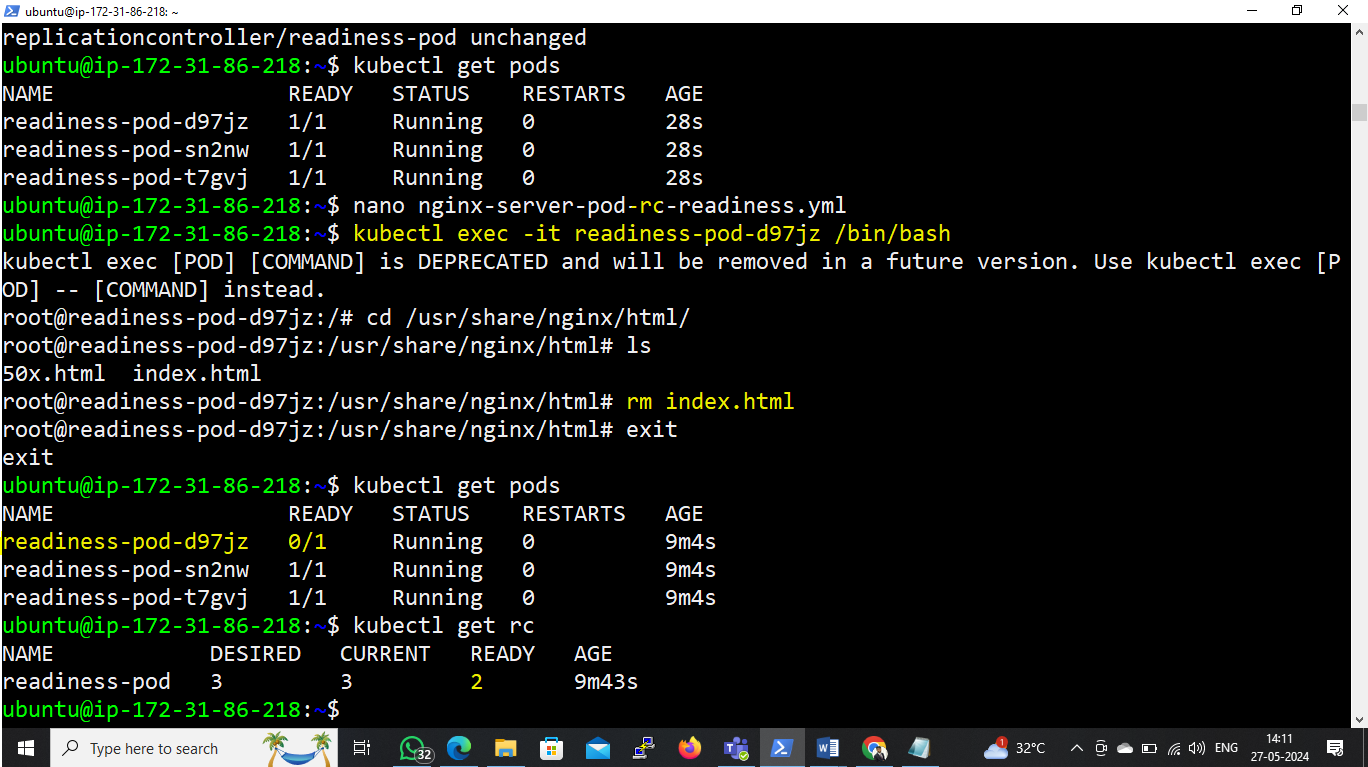
* kubectl apply -f nginx-server-pod-rc-readiness.yml
* kubectl get pods



# Enter in pod **readiness-pod-d97jz**

* kubectl exec -it readiness-pod-d97jz /bin/bash

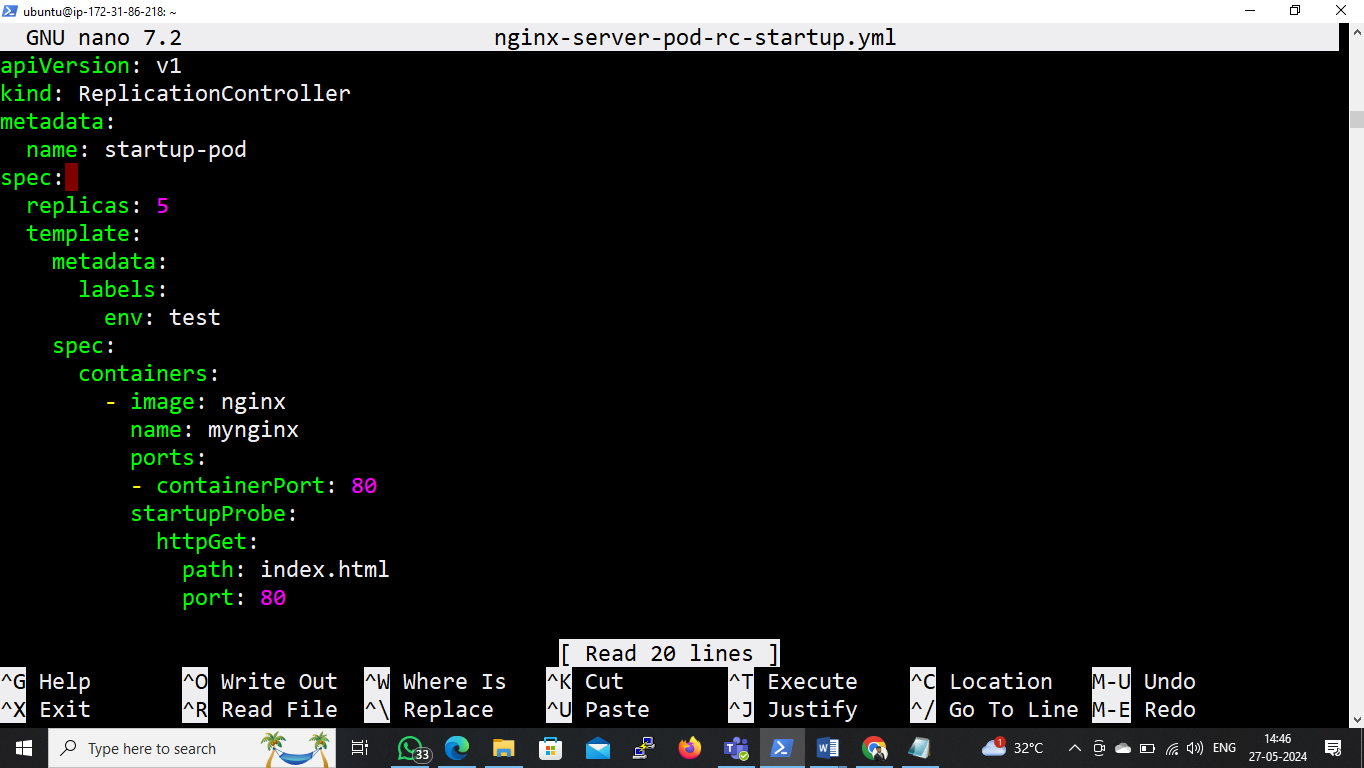
**# Remove index.html page and when probe check health of pods they will get un-healthy, and mark as not ready which shows in below screenshot;**



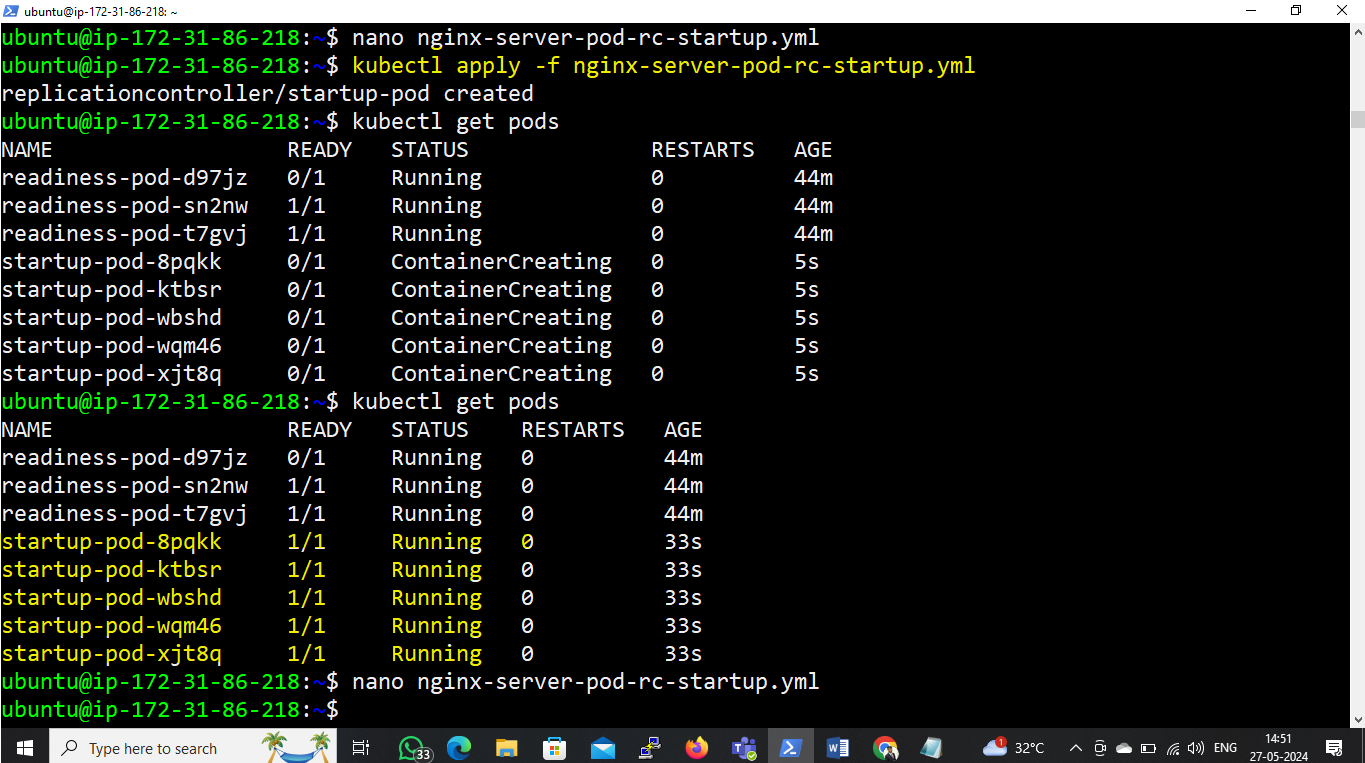
**Use of StartupProbe:**

* **Startup Probe:**
* **Purpose:** Determines whether the container in a Pod has started successfully.
* **Functionality:** Runs probes during the initial startup of the container, delaying the readiness check until the application inside the container has started.
* **Typical Use Case:** Used for applications with a long startup time or complex initialization process. It allows Kubernetes to wait until the application is fully up and running before sending traffic to the Pod.
* **Action on Failure:** If the startup probe fails, the Pod is treated as failed, similar to how a liveness probe failure is handled.

# Create a manifestfile of nginx-server-pod-rc-startup.yml



* kubectl apply -f nginx-server-pod-rc-startup.yml
* kubectl get pods



* In summary, liveness probes ensure that the **container is running correctly**, readiness probes ensure that the **container is ready to serve traffic**, and startup probes **delay the readiness check until the container has started successfully**.
* Each probe type serves a specific purpose in managing the lifecycle and health of applications running in Kubernetes Pods.