### Methodology

To deploy images using Kubernetes in Pods, ReplicaSets, and Deployments on AWS (Amazon Web Services) and GCP (Google Cloud Platform), you’ll need to follow several steps. Here’s a high-level overview of the process:

##### Set up a Kubernetes cluster:

On both AWS and GCP, you need to create a Kubernetes cluster where you’ll deploy your applications. The process may vary slightly between platforms, but both AWS and GCP offer managed Kubernetes services (Amazon Elastic Kubernetes Service – EKS on AWS, Google Kubernetes Engine – GKE on GCP) that simplify cluster creation.

##### Containerize your application:

Before deploying your application, you need to package it into a container image. You can use Docker to create a Dockerfile that describes the steps needed to build the container image.

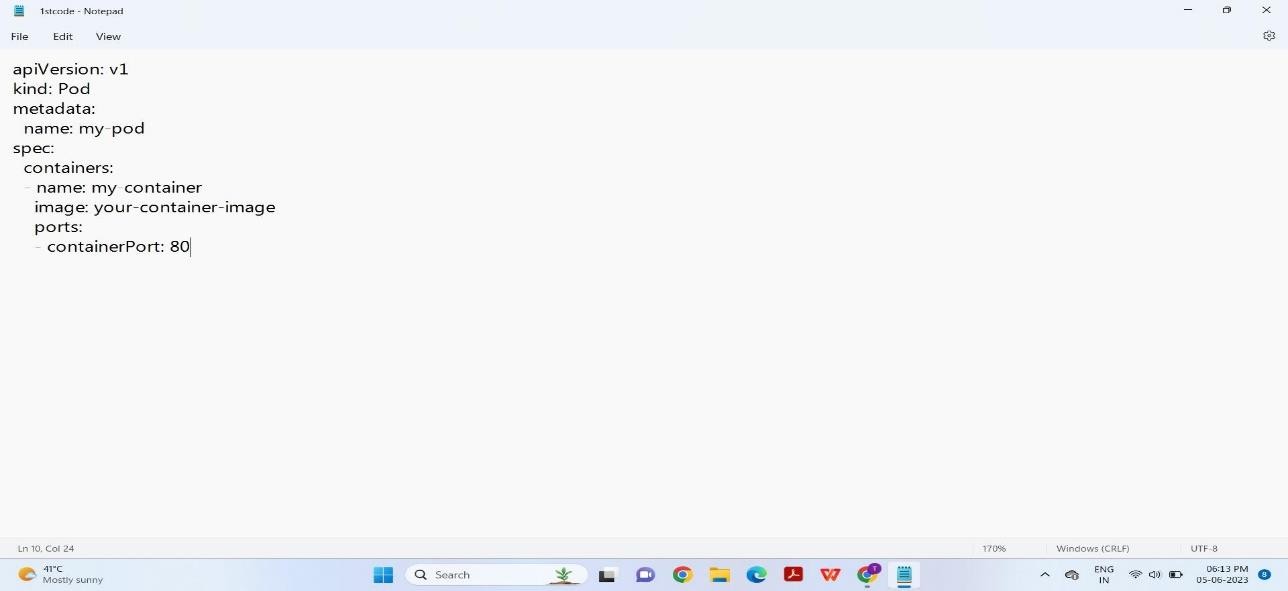
##### Build and push the container image:

Build the container image using Docker and then push it to a container registry. AWS has Amazon Elastic Container Registry (ECR), while GCP has Google Container Registry (GCR). Pushing the image makes it available for deployment within your cluster.

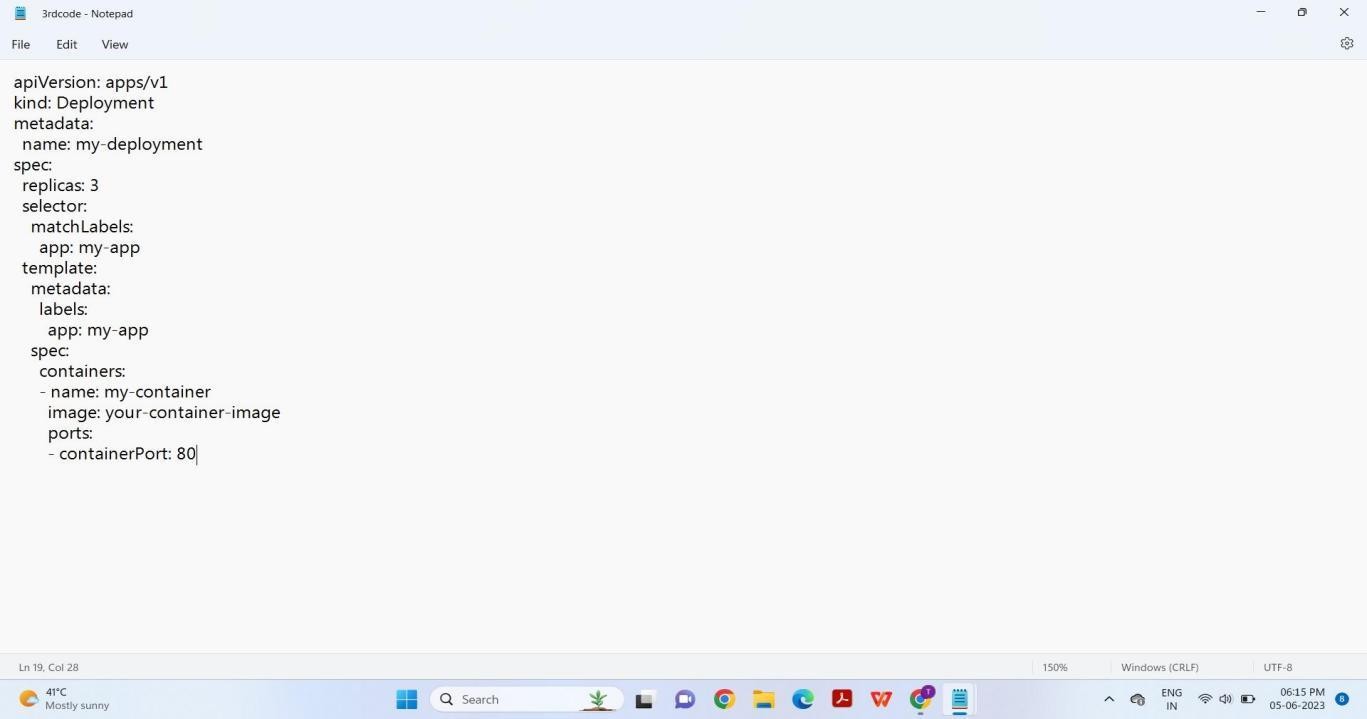
##### Define Kubernetes manifests:

Create YAML or JSON manifests that describe the desired state of your application within Kubernetes. You’ll typically define Pods, ReplicaSets, and Deployments in these manifests.

* + - **Pods :** Pods are the smallest deployable units in Kubernetes. They encapsulate one or more containers (your application) and associated resources like storage volumes or network configurations. Define a Pod manifest that references the container image you pushed to the container registry.



* + - **Replicaset :** ReplicaSets ensure a specified number of Pod replicas are running at any given time. They help maintain the desired level of availability and scale your application. Define a ReplicaSet manifest that specifies the desired number of replicas and references the Pod manifest.
    - **Deployments :** Deployments provide declarative updates for Pods and ReplicaSets, allowing you to manage application rollouts and rollbacks. Define a Deployment manifest that references the ReplicaSet manifest and specifies update strategies, scaling policies, and other deployment-specific configurations.



##### Deploy the manifests:

Use the Kubernetes command-line tool (kubectl) to deploy the manifests to your Kubernetes cluster. This will create the Pods, ReplicaSets, and Deployments based on the definitions you provided.

* + - **AWS :** If you’re using EKS, configure the kubectl command to connect to your EKS cluster by following the official documentation.
    - **GCP:** If you’re using GKE, configure the kubectl command to connect to your GKE cluster by following the official documentation.

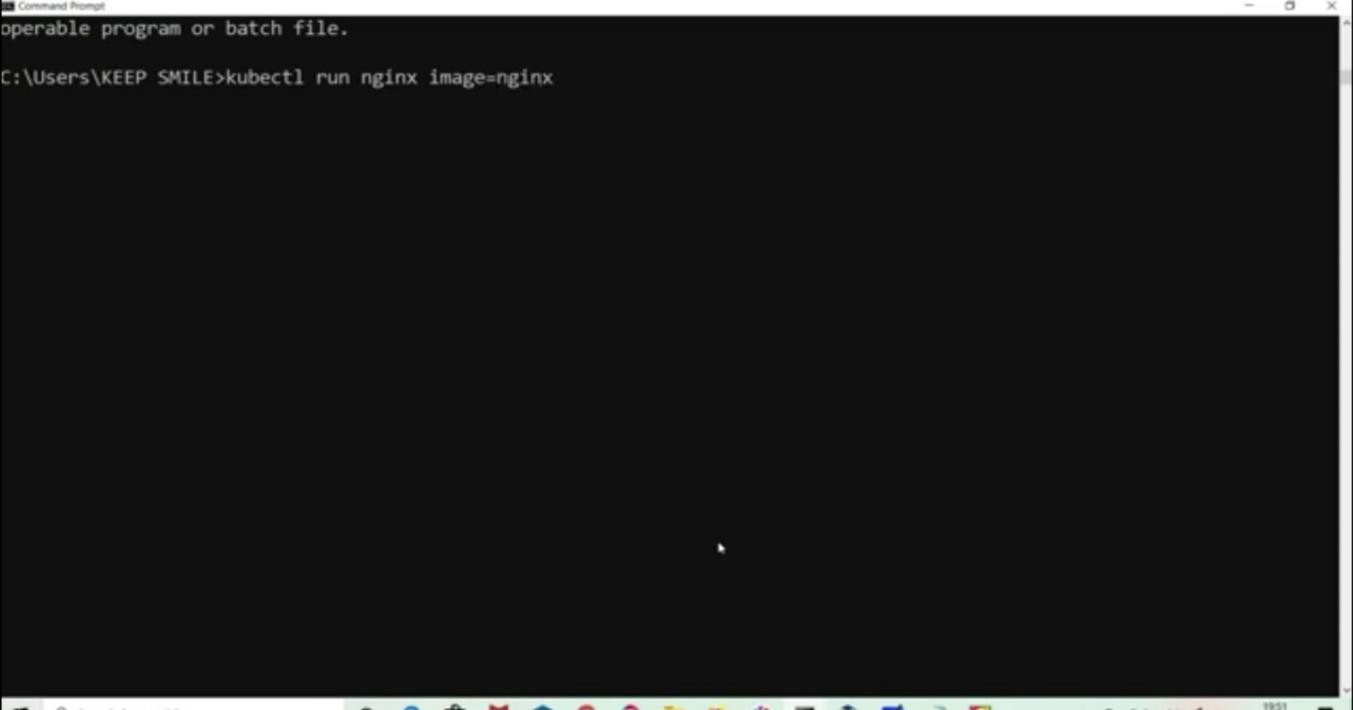
##### Monitor and manage the deployment:

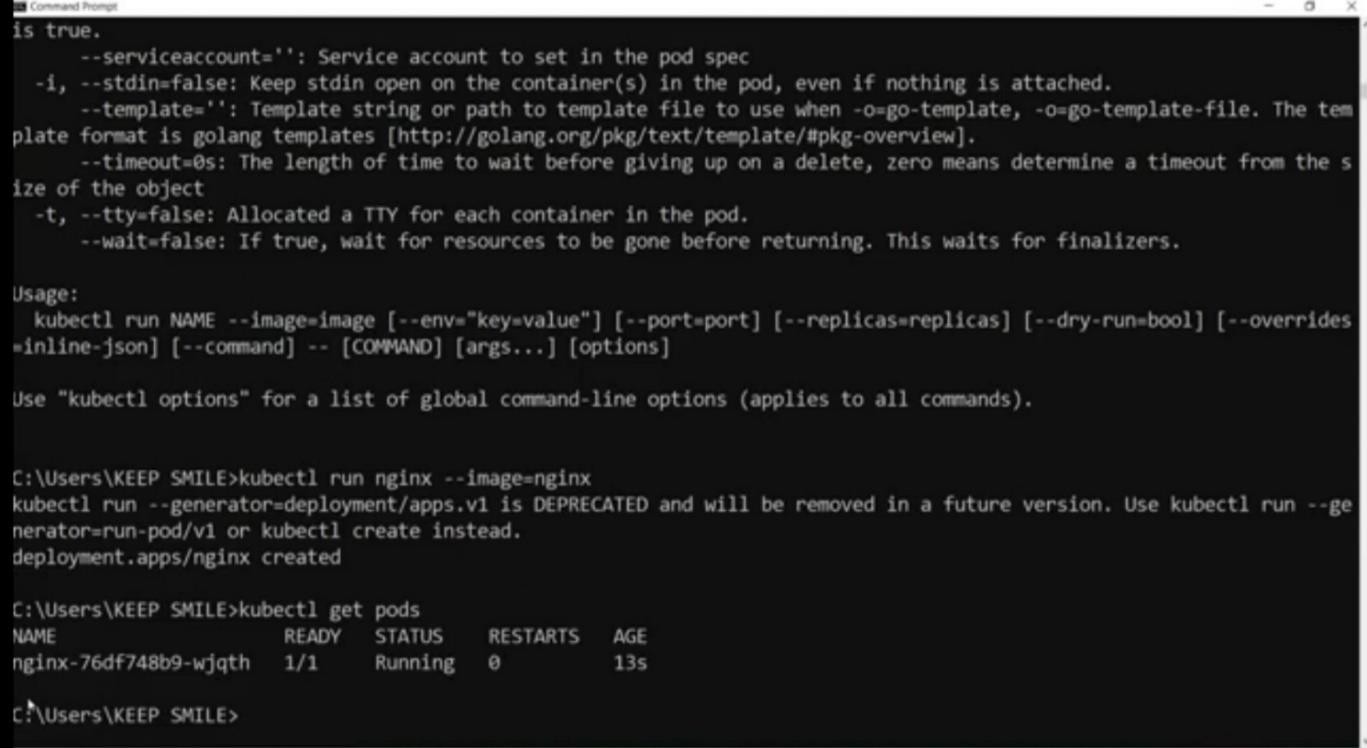
Once your application is deployed, you can use Kubernetes management features to monitor and manage your deployment. This includes scaling the number of replicas, updating the image version, rolling out new releases, and troubleshooting any issues that may arise.

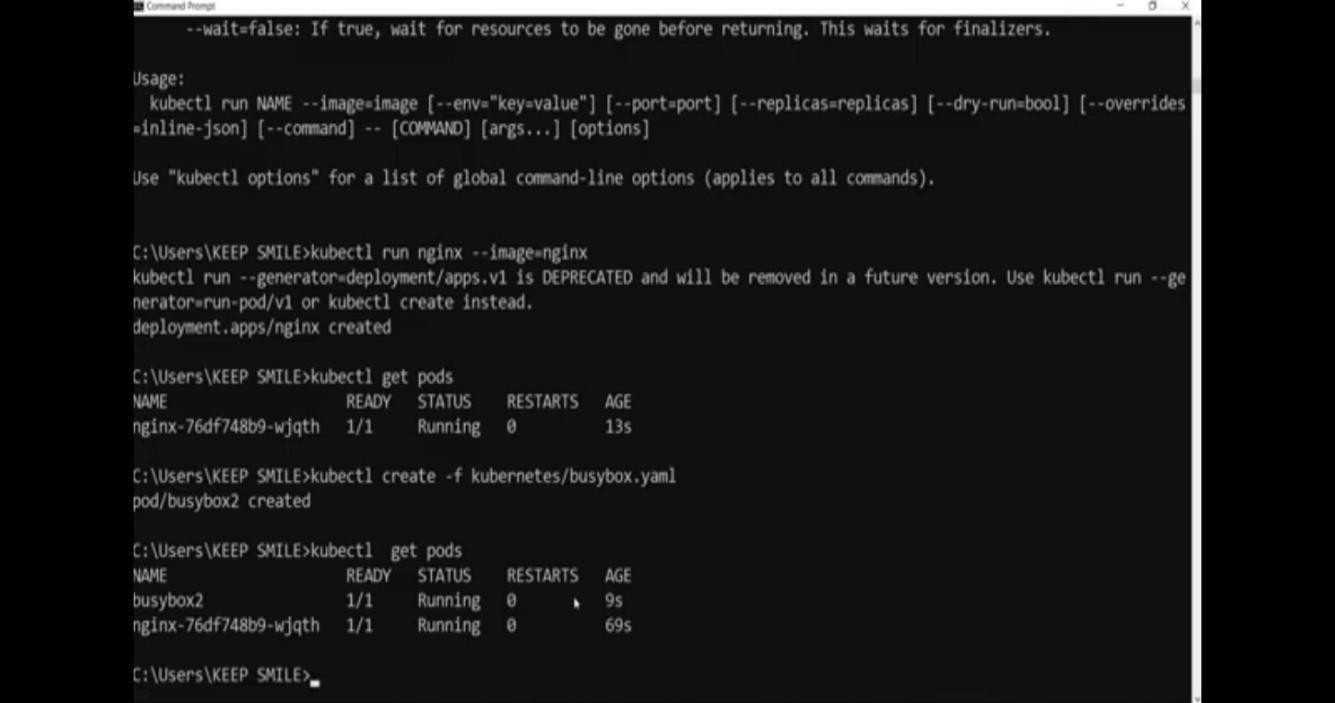
## RESULT

##### Step-1 : Define the pods

* Pods are the smallest unit in Kubernetes and represent a single instance of a running process.
* You can define a Pod manifest file specifying the container image, ports, environment variables, and volumes.

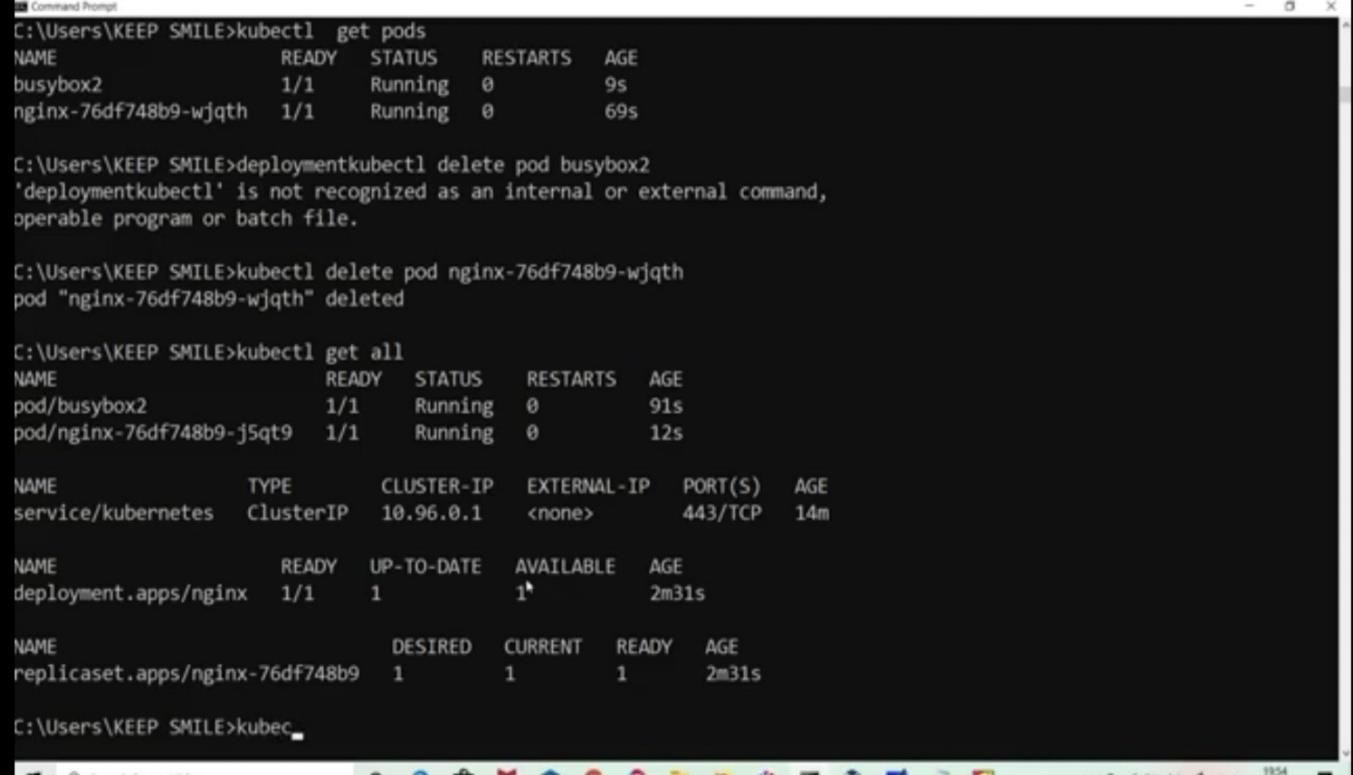


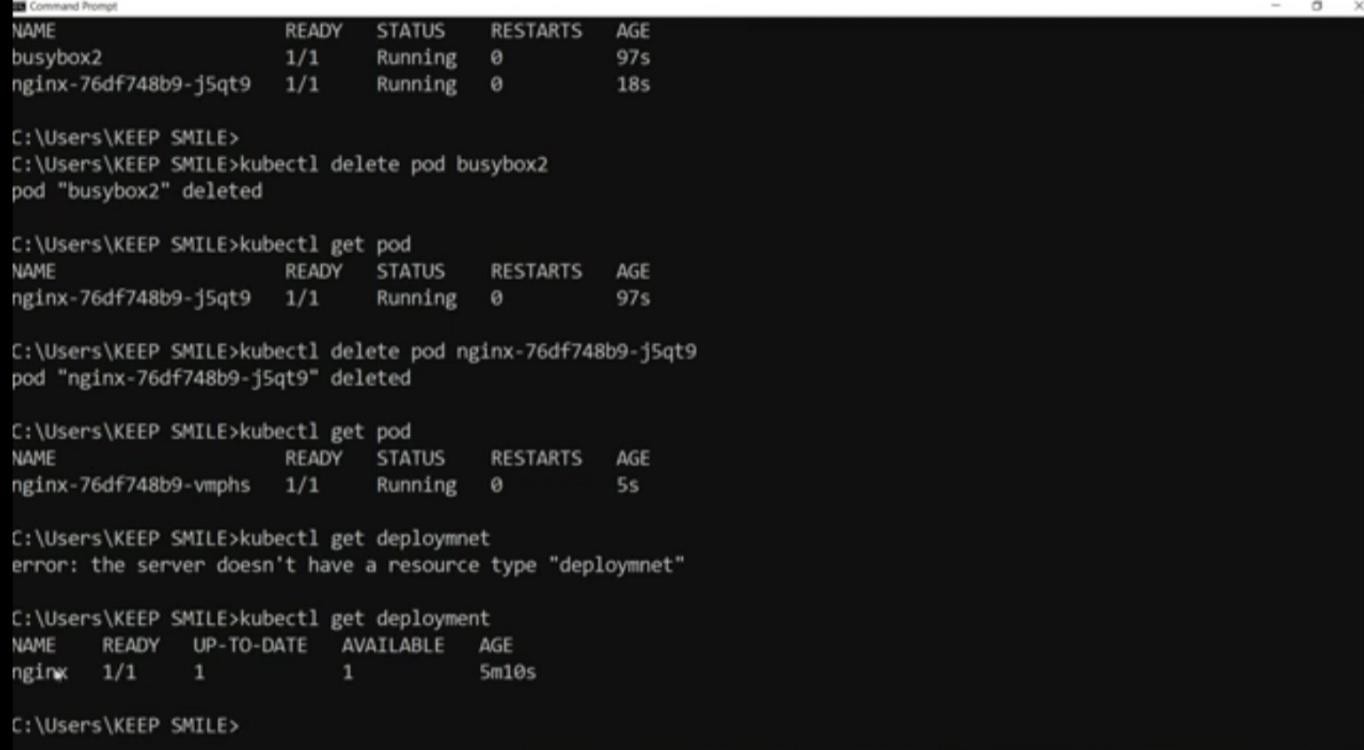




##### Step-2 : Define a deployment manifest file.

* Deployments are higher-level abstractions built on top of ReplicaSets.
* They offer declarative updates, making it easier to manage rolling updates, scaling, and rollbacks of your application.
* You define a Deployment manifest file similar to a ReplicaSet but with additional deployment-specific options.





##### Step-3 : Define a ReplicaSet manifest file.

* ReplicaSets ensure a specified number of replicas (Pods) are always running in the cluster.
* They provide high availability and scalability by automatically managing Pod creation, scaling, and deletion.
* You define a ReplicaSet manifest file that specifies the desired number of replicas, selector labels, and Pod template.

