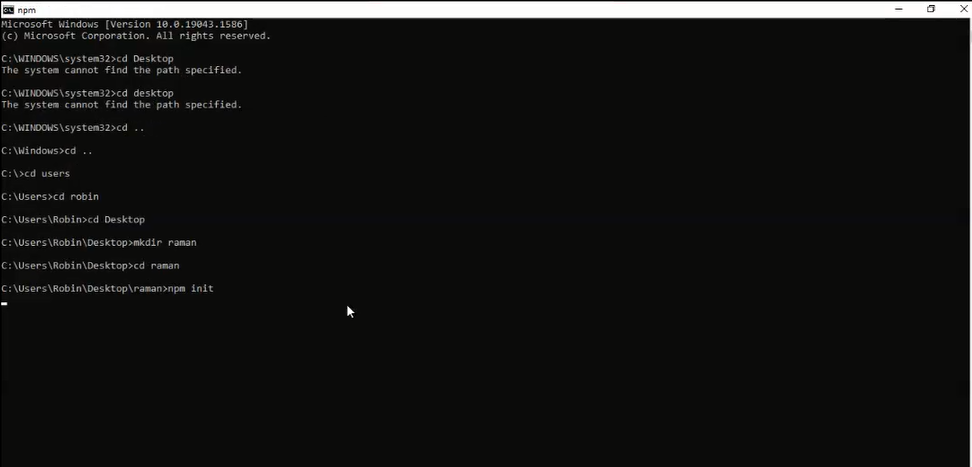
**Pre-Requisites: Docker, Kubectl, Latest version of Cloud CLI, Kubernetes cluster on IBM Cloud.**

**Steps:**

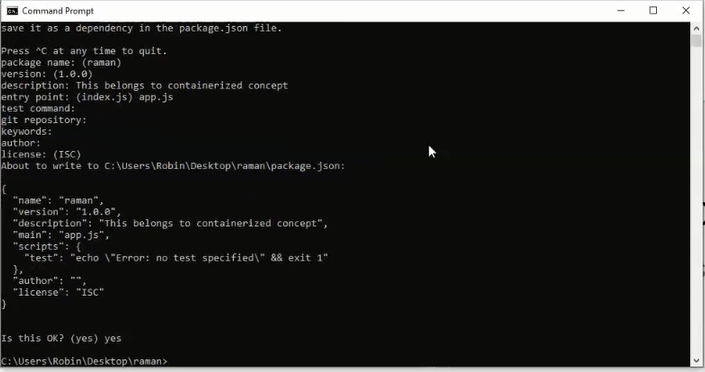
**Create a Node js application**

**Steps:**

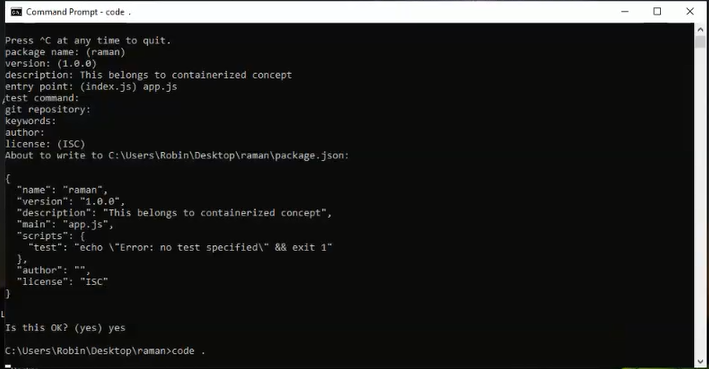
1. **Open cmd**
2. **To create basic node js project make dir using command mkdir raman on desktop**
3. **Go inside this folder and initialize npm using command npm init**

****

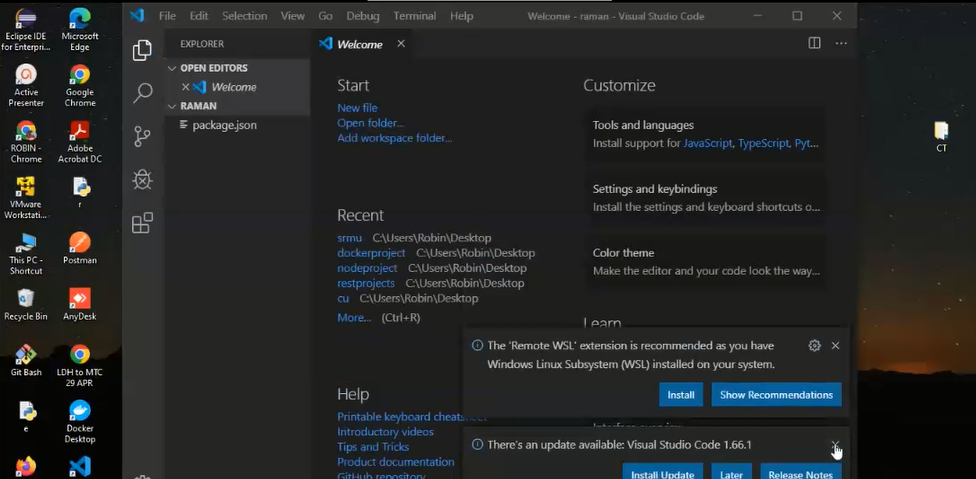
1. **Initialize the settings for package.json**

****

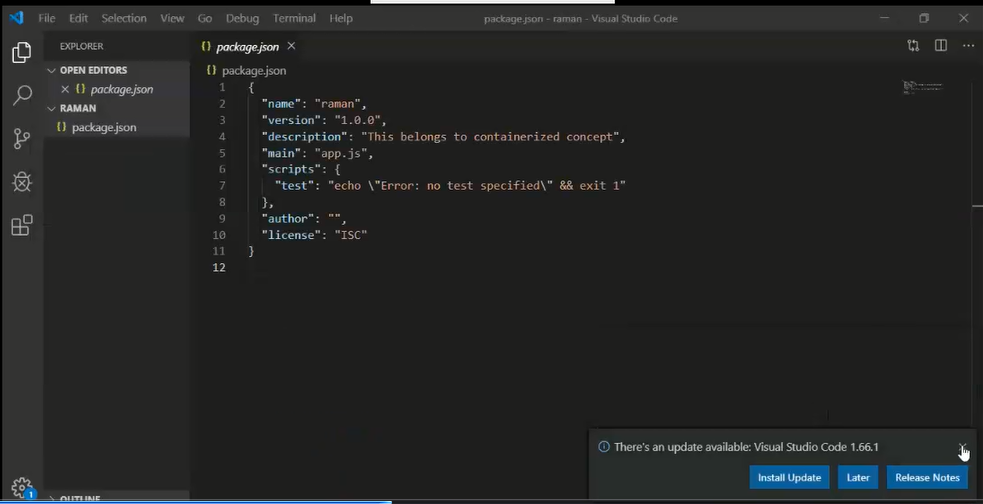
1. **To open the code in visual studio give command code .**

****

1. **Open Visual Studio**

****

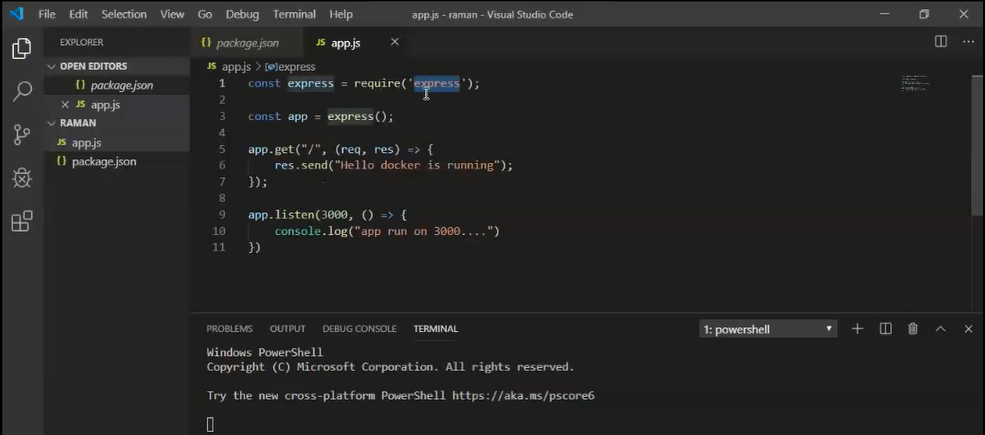
1. **Open package.json**

****

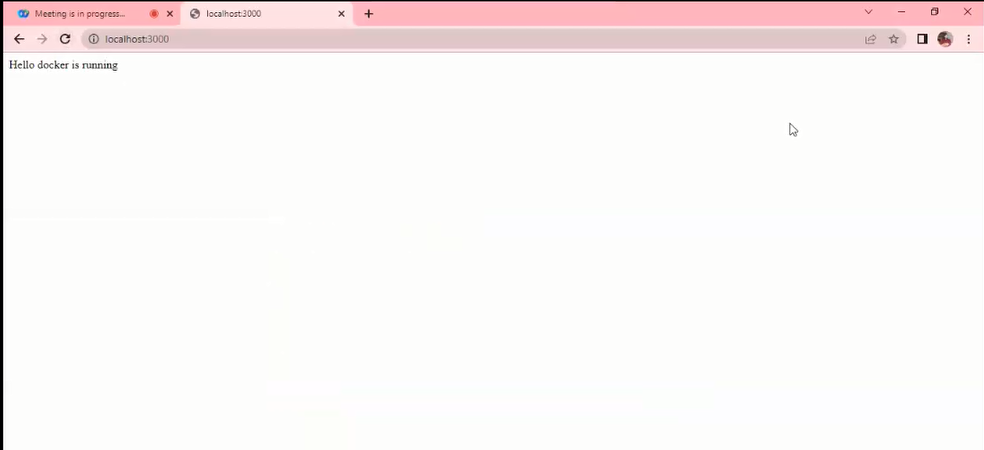
1. **Open terminal and create new file through command or create new file through option available inside raman folder**

****

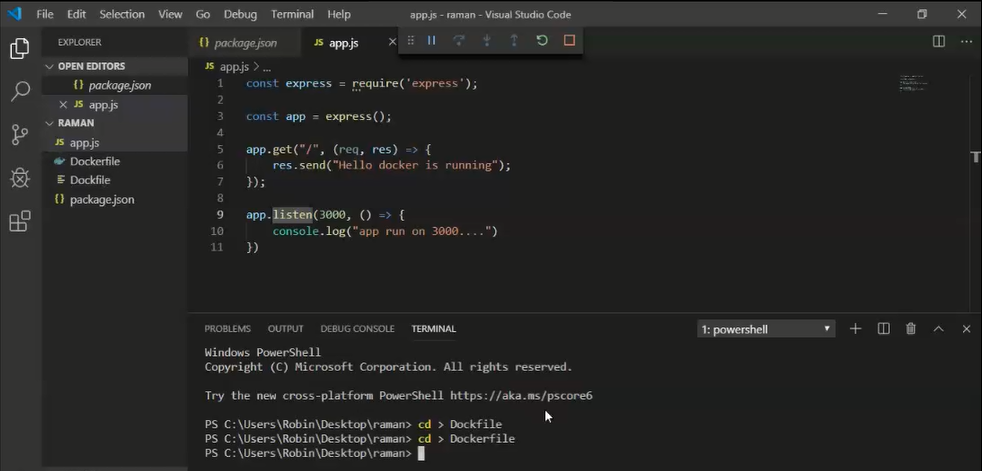
1. **Put sample express code in app.js**

****

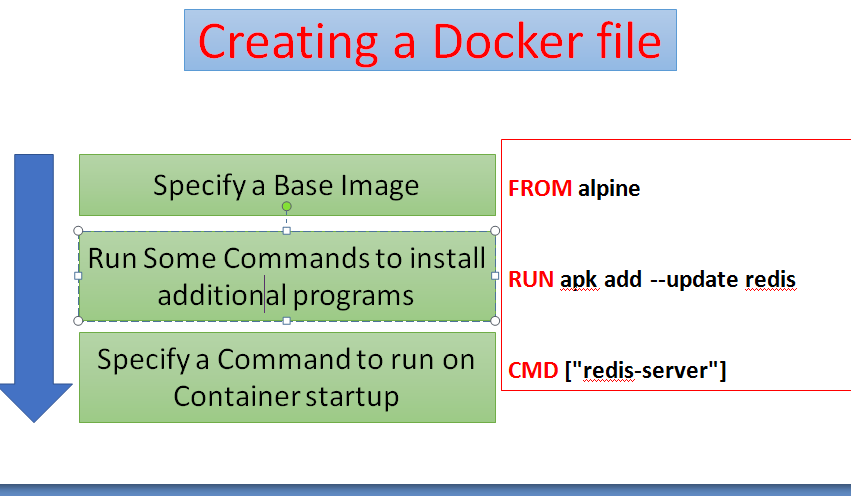
1. **Test sample app without container concept initially**

****

1. **Create Dockerfile**

****

1. **Example Dockerfile**

****

**Dockerfile commands to create image from app.js**

**FROM node:12-alpine**

**#RUN command to make sure to grant necessary permission into node\_module and app folders, so that when npm install can be executed.**

**RUN mkdir -p /home/node/app/node\_modules && chown -R node:node /home/node/app**

**#WORKDIR command to set working directory**

**WORKDIR /home/node/app**

**#COPY command to copy package.json file.**

**COPY package\*.json ./**

**USER node**

**RUN npm install**

**#Copy your application code with the appropriate permissions to the application directory on the container:**

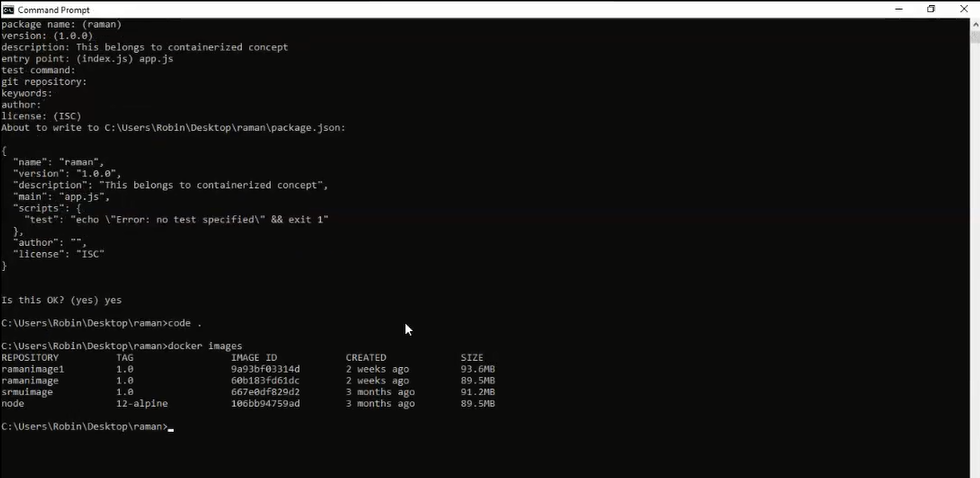
**COPY --chown=node:node . .**

**#EXPOSE command to expose port**

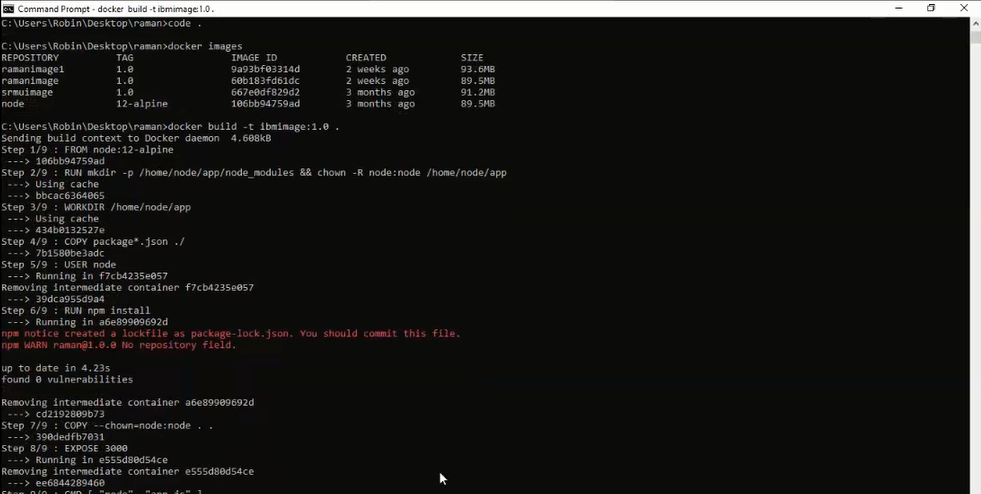
**EXPOSE 3000**

**CMD [ "node", "app.js" ]**

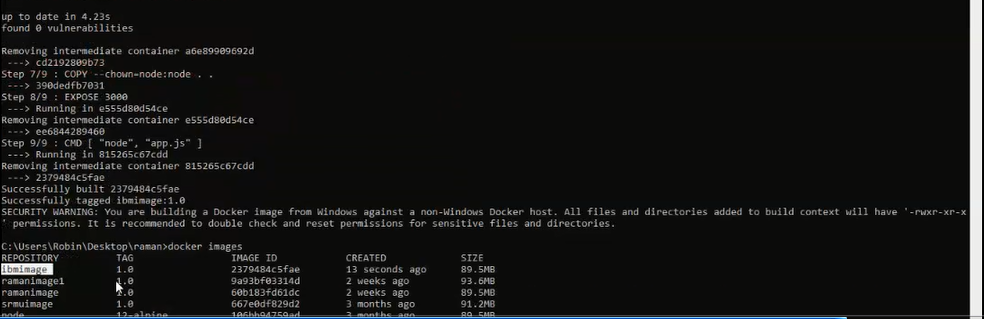
1. **Run command to check for present docker images**

****

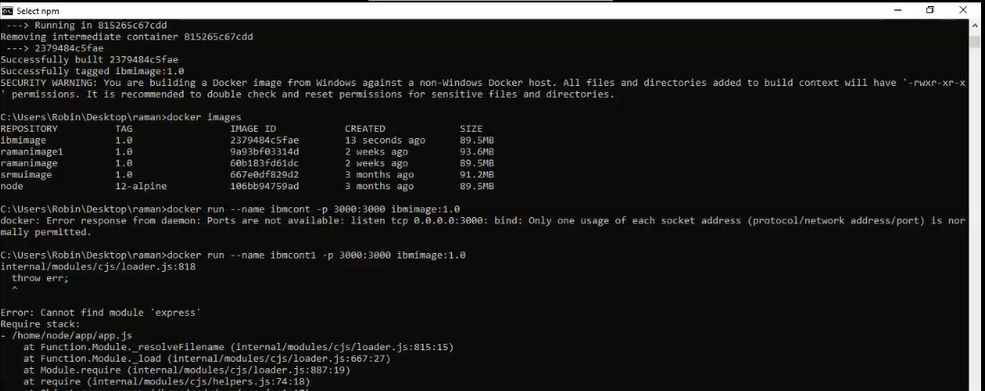
1. **Run command to create an image using docker file**

****

**15.)Again check if the image is created**

****

1. **Create container for the image file to deploy it**

****

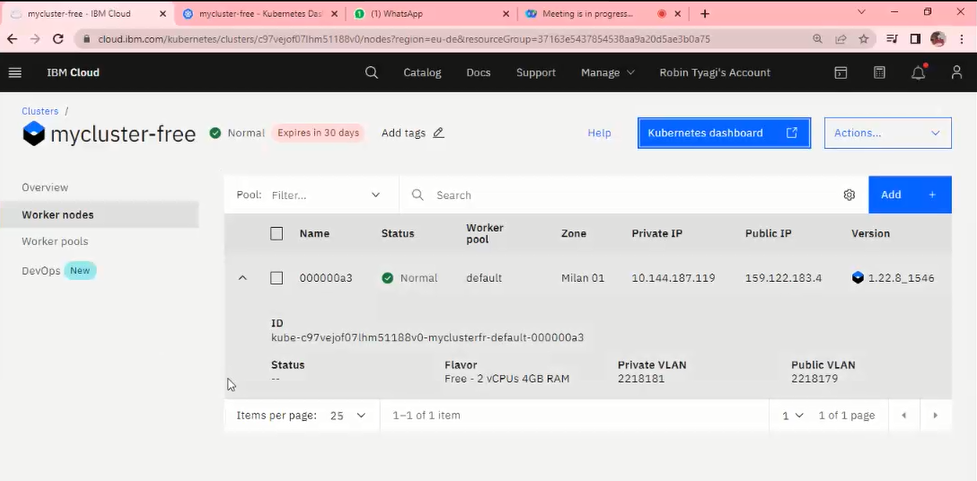
1. **Check for the container**

****

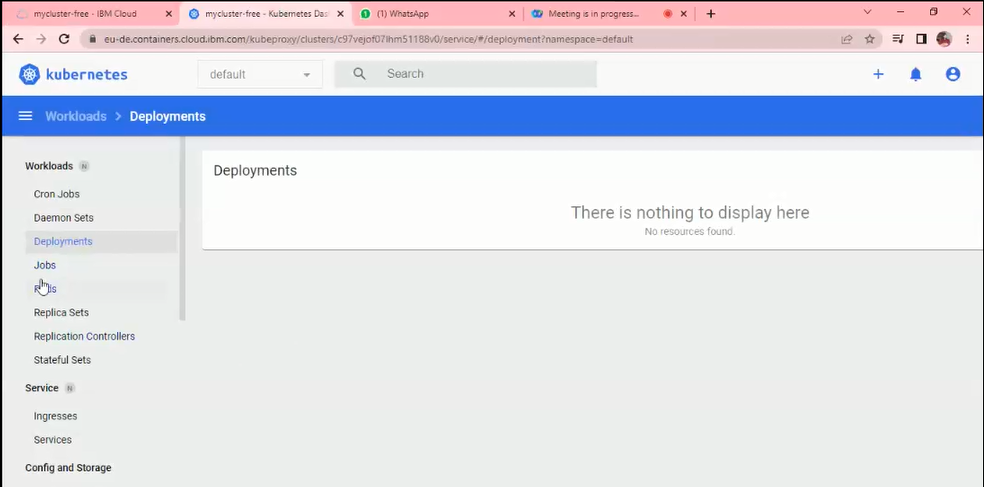
**17.)**

**Now deploy this containerized app using kubernetes**

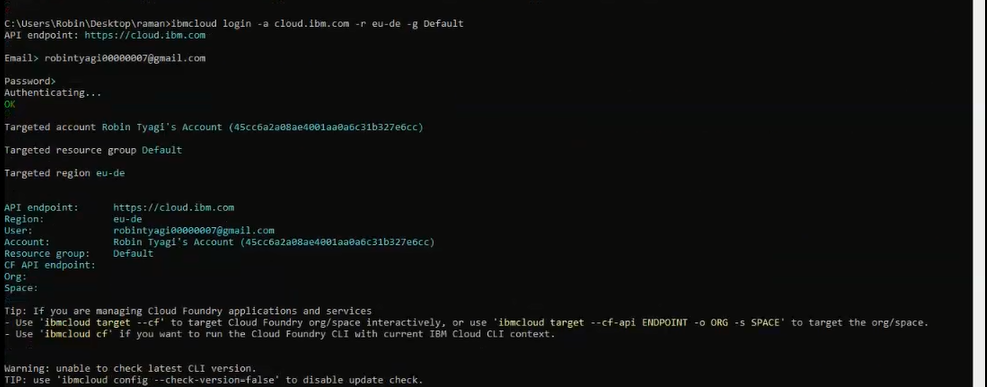
**For this we need to create a kubernetes cluster in lite mode on ibm cloud. This service takes 15-20 minutes to create**

****

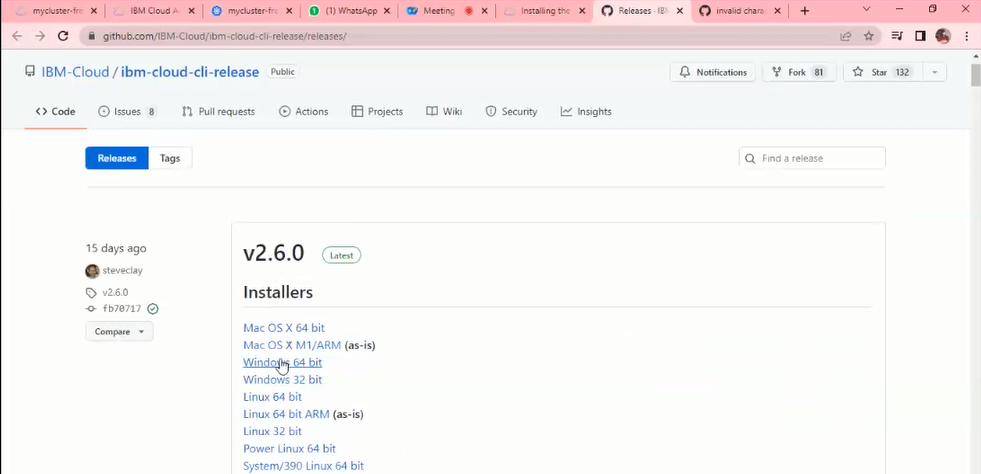
1. **Click on kubernetes dashboard**

****

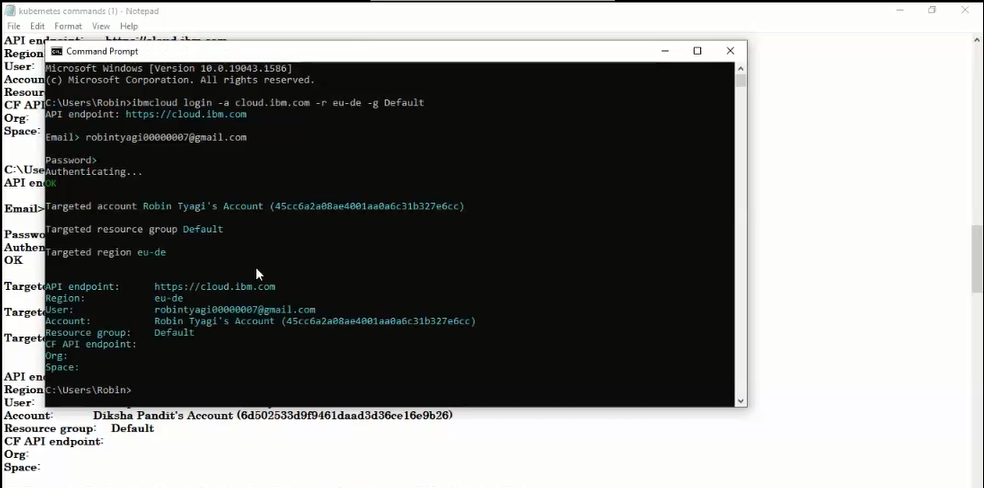
1. **First we need to push our data to container registry. For that login to ibmcloud setting target region to the region in which kubernetes service runs**

****

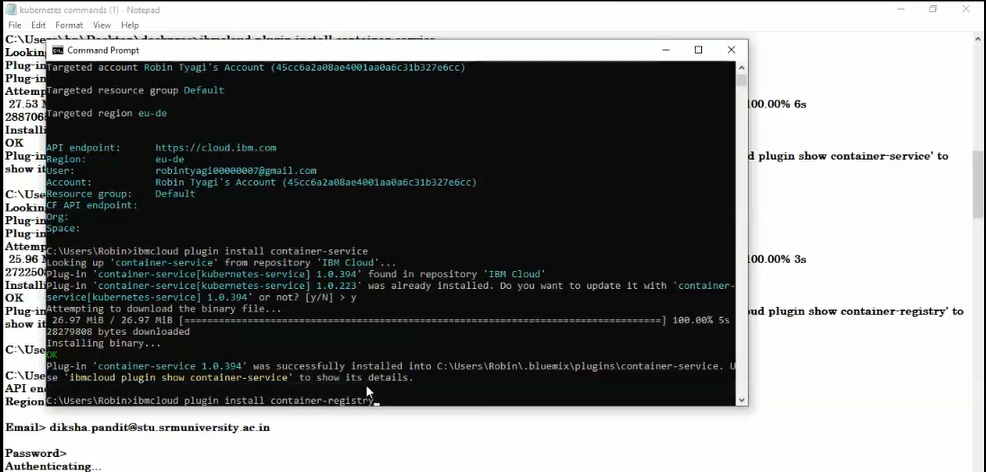
1. **Before giving any other command, keep a check that ibm cloud CLI version should be latest.(v2.6.0)**

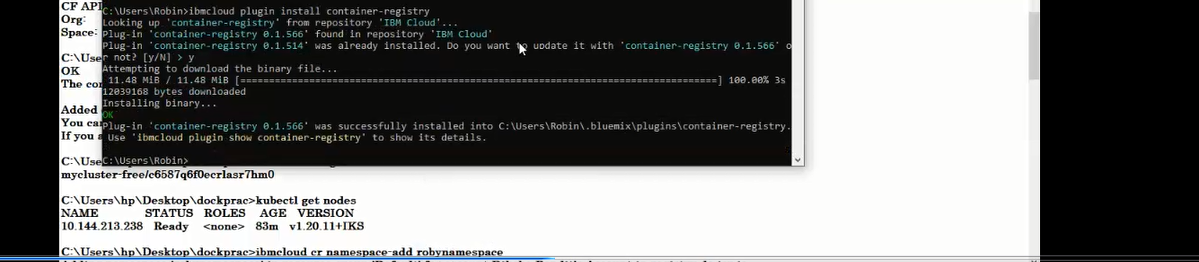
****

1. **After installing latest version, reopen cmd and login to cloud with target region and resource group defined.**

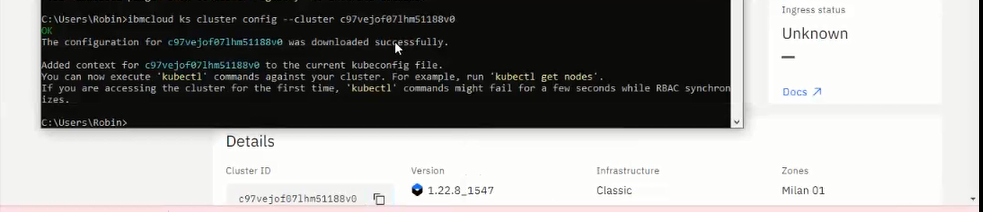
****

1. **Install the required plugins to work with container registry and kubernetes service**

****

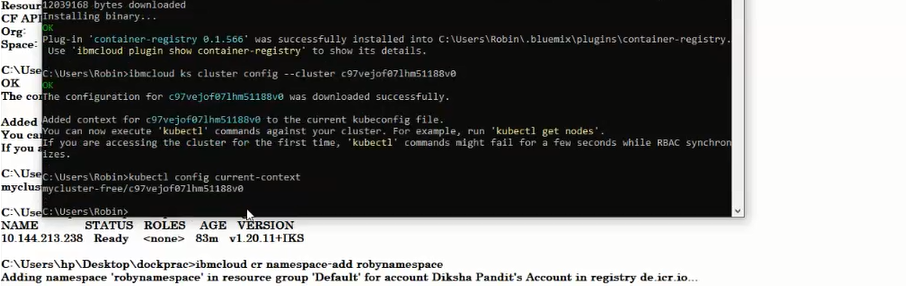
****

1. **Download kubernetes cluster configuration (for this we need cluster ID), WE get once create service on cloud**

****

1. **To run next commands we need to have kubectl in your system**

**Run command to check configuration of cluster with which we are going to work**

****

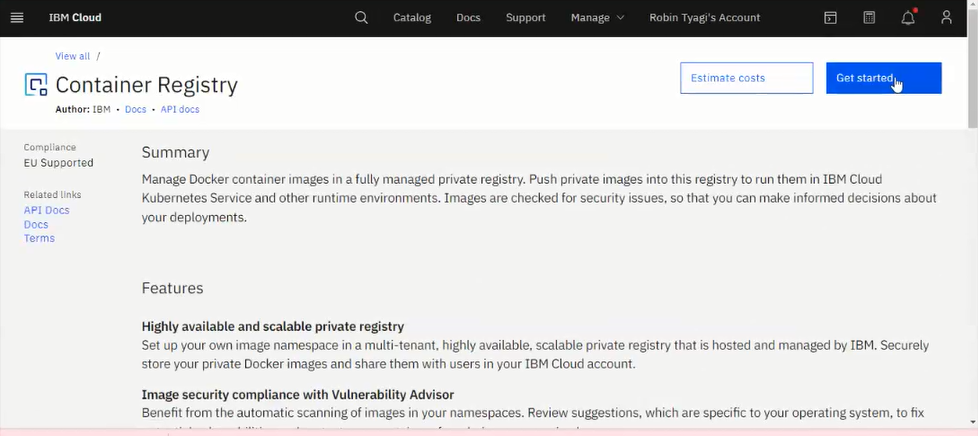
1. **To check worker nodes run command kubectl get nodes(if required ), not necessary to run now**
2. **Now to create namespace in container registry run the command(to register container)**

****

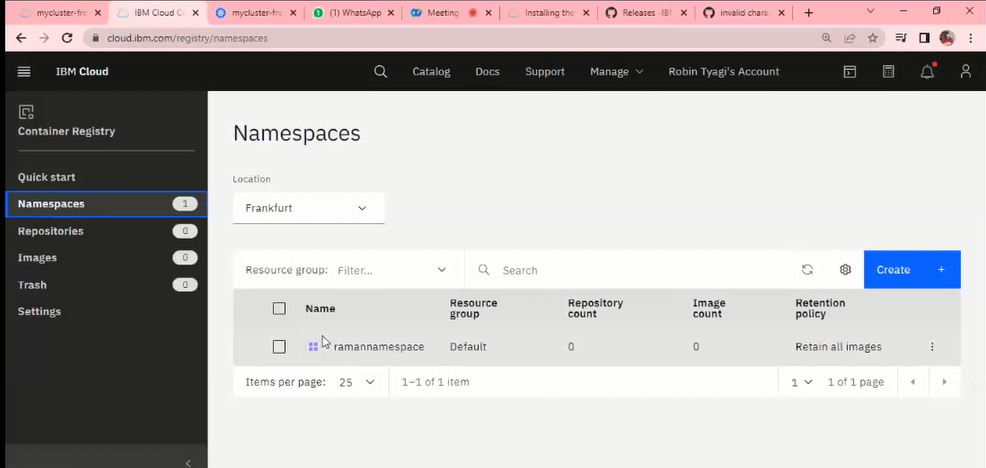
1. **Then login to container registry and check if namespace you created is added or not**

****

1. **To cross check if namespace is created, go to ibmcloud, click on catalog, go to container registry and click on get started**

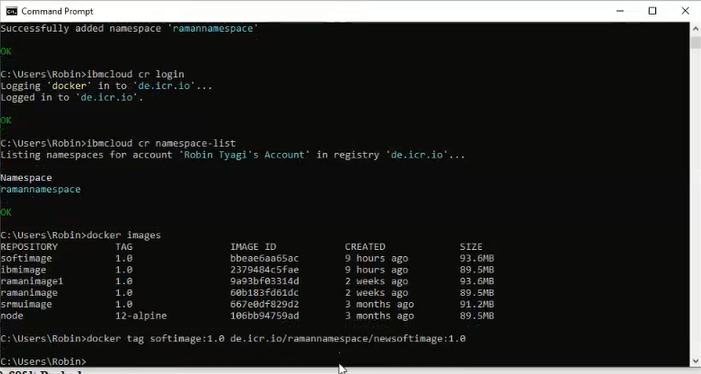
****

1. **Check on namespace option and choose region Frankfurt**

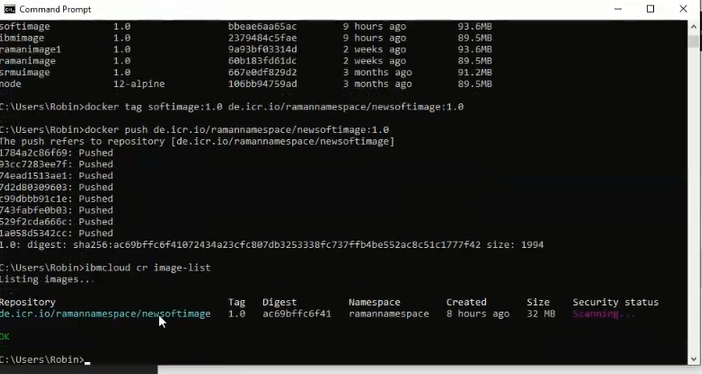
****

1. **Now tag your image and push it on container registry**

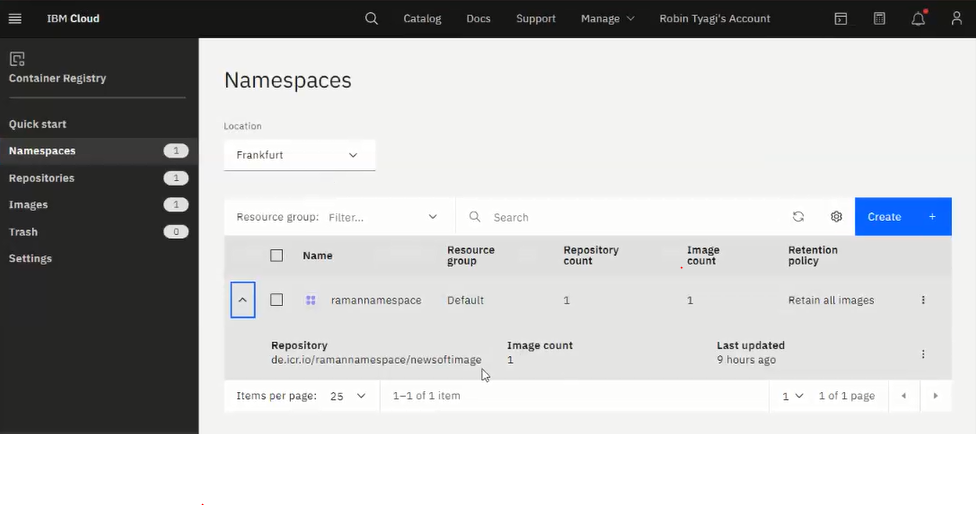
**In command replace your image name , change namespace to name you created and give tag name to image**

****

1. **Now push tagged image to container registry**

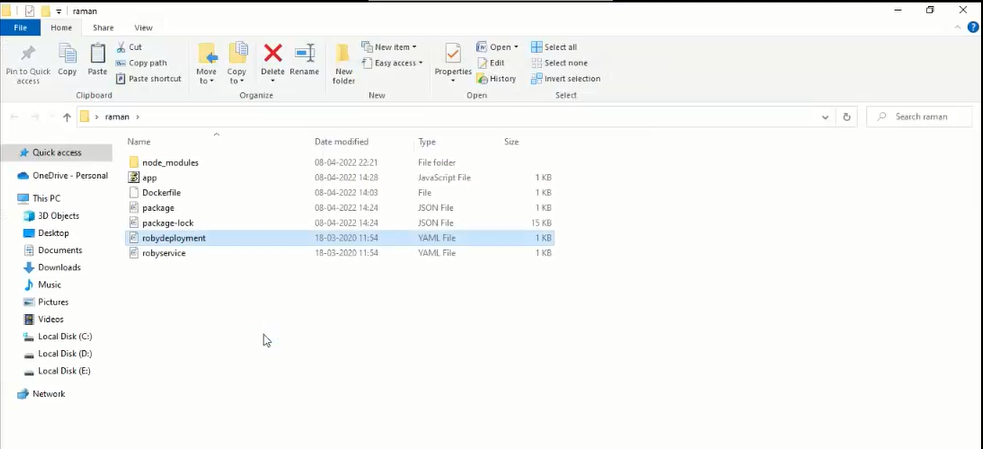
****

1. **Refresh container registry to check for images**

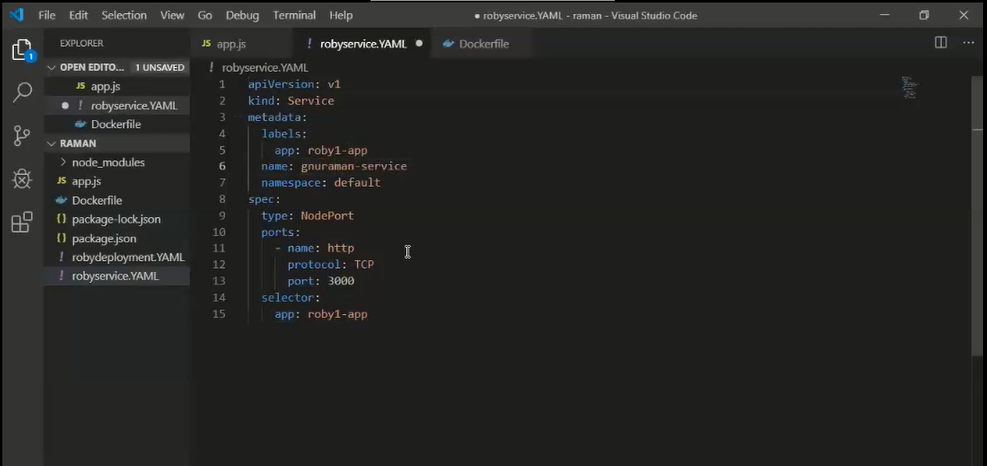
****

1. **Now create two files i.e service file and deployment file(service file creates the settings to generate external point to run application and deployment file decides number of replicas and other settings related to custer)**

**Place the files provided in the folder where the application is:**

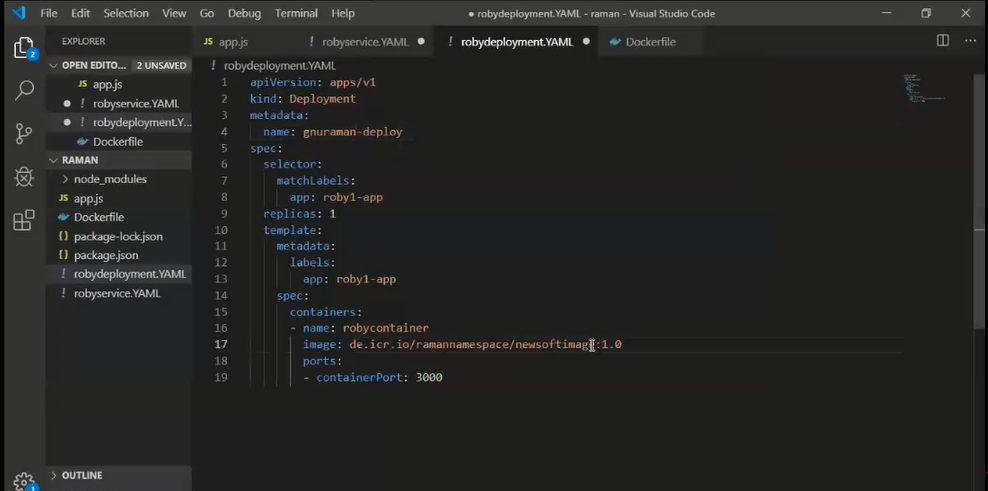
****

1. **Make changes to the service.yaml file like change the service file name,**

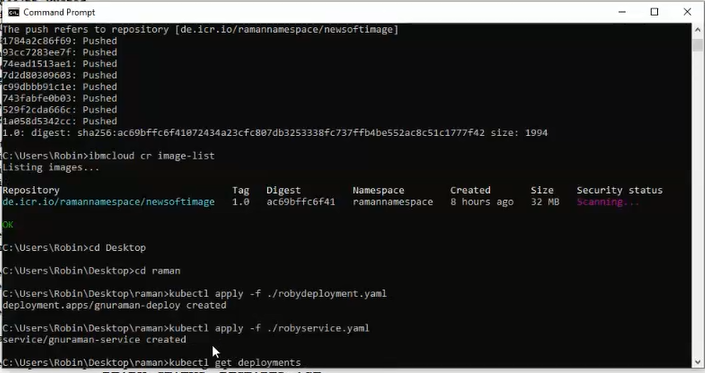
****

1. **Make changes to deployment file**

**Change the name in metadata, keep it same as service file name and put the location of your image file, also can change the number of replicas if you want.**

****

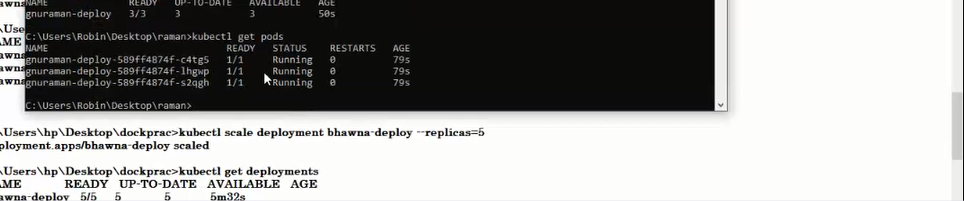
1. **Go to the folder, where main application resides. And run commands to apply the file changes you made.**

****

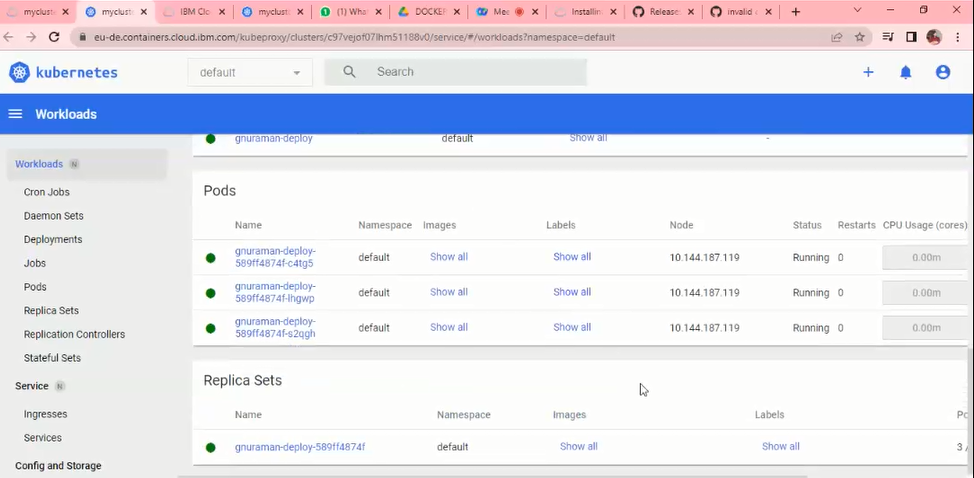
1. **Run command to check deployments**

****

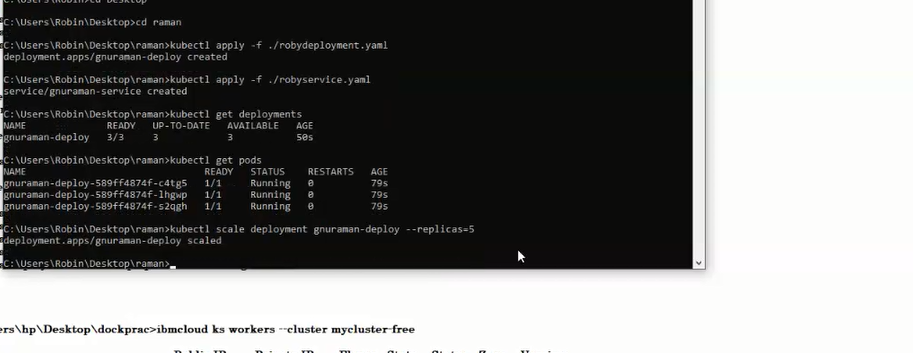
1. **Check pods**

****

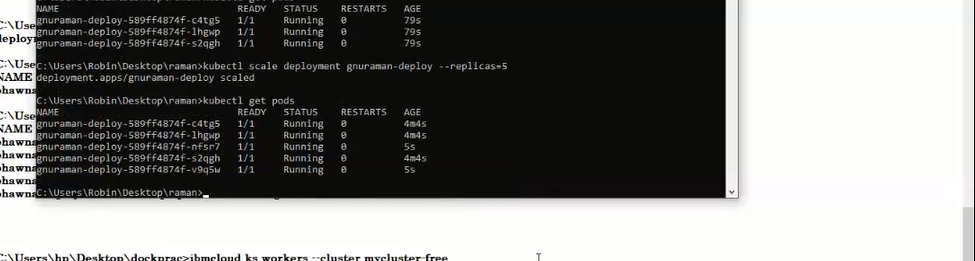
1. **To check for the pods go to kubernetes dashboard**

****

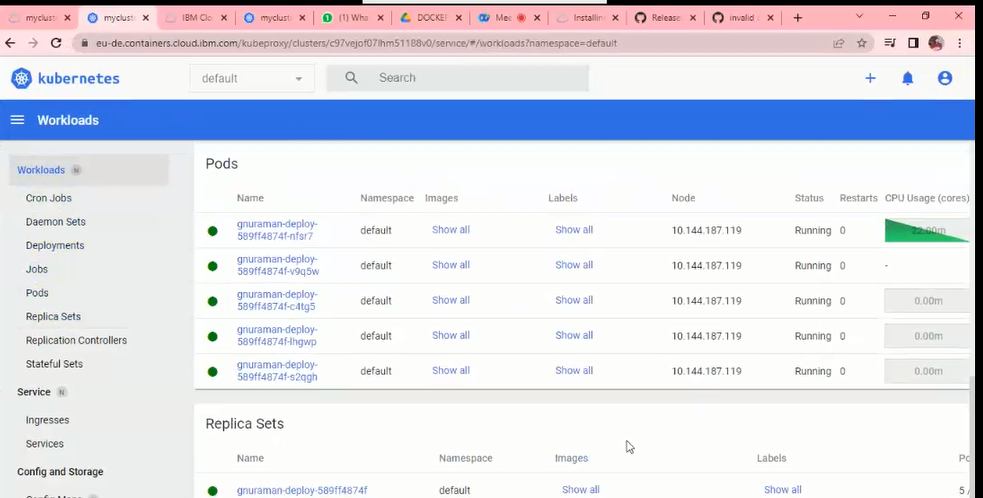
1. **If you want to increase replica, then run the following command, remember to change file name to the name kept in deployment file(i.e gnuraman)**

****

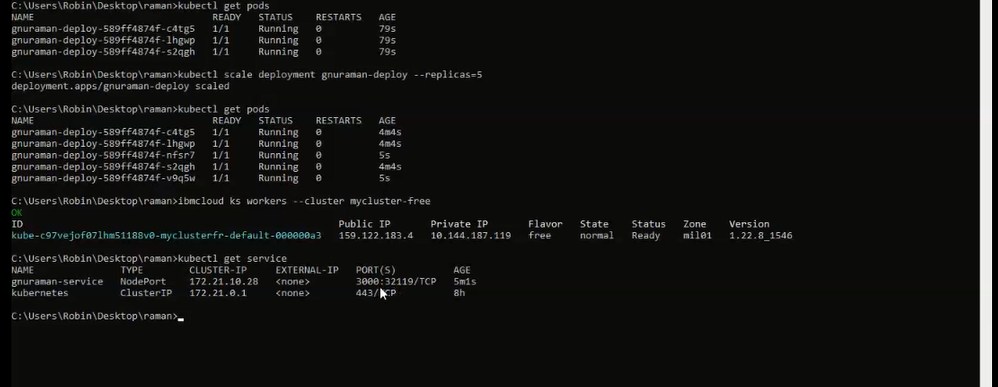
1. **Check number of pods**

****

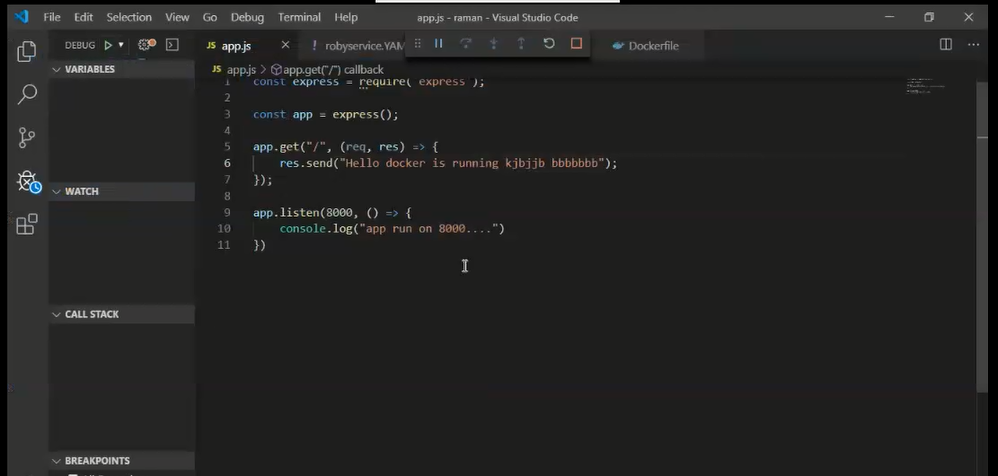
1. **Chck on kubernetes as well**

****

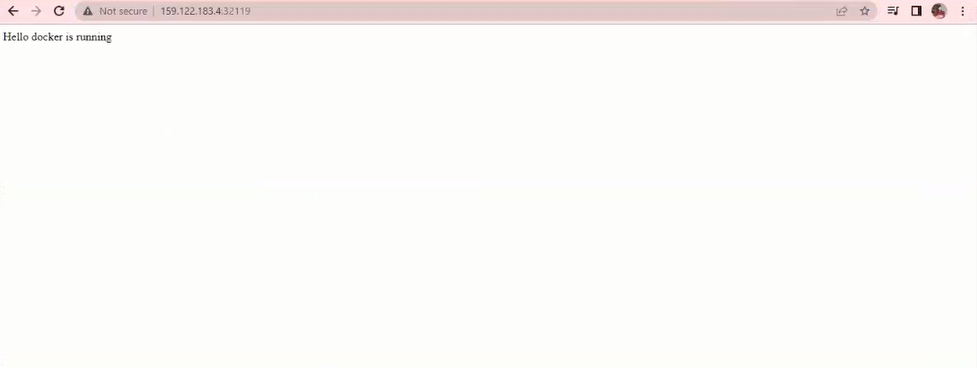
1. **To run app on external point , run command to check external ip and port**

****

1. **Now make changes to the code and observe if changes reflect on the external ip you use to run your app.**

****

1. **Now run the app on visual studio an also run the app using ip received for external service**
2. **Use public ip to run the app and port assigned**

****

1. **Also check the self healing property by deleting PODS and check it gets healed automatically.**