# HELM CHARTS (package manager for Kubernetes)

Contents

[HELM CHARTS (package manager for Kubernetes) 1](#_Toc72946291)

[What is Helm? 1](#_Toc72946292)

[Contents of a Helm Chart 2](#_Toc72946293)

[Install Helm on windows 2](#_Toc72946294)

[Creating Helm chart 3](#_Toc72946295)

[Upgrading Helm Chart 6](#_Toc72946296)

[Rollback Helm Chart to earlier versions 6](#_Toc72946297)

[Uploading Helm Chart to registry 7](#_Toc72946298)

## What is Helm?

[Helm](https://helm.sh/) is widely known as "the package manager for [Kubernetes](https://azure.microsoft.com/services/kubernetes-service/?WT.mc_id=containers-19838-ludossan)".

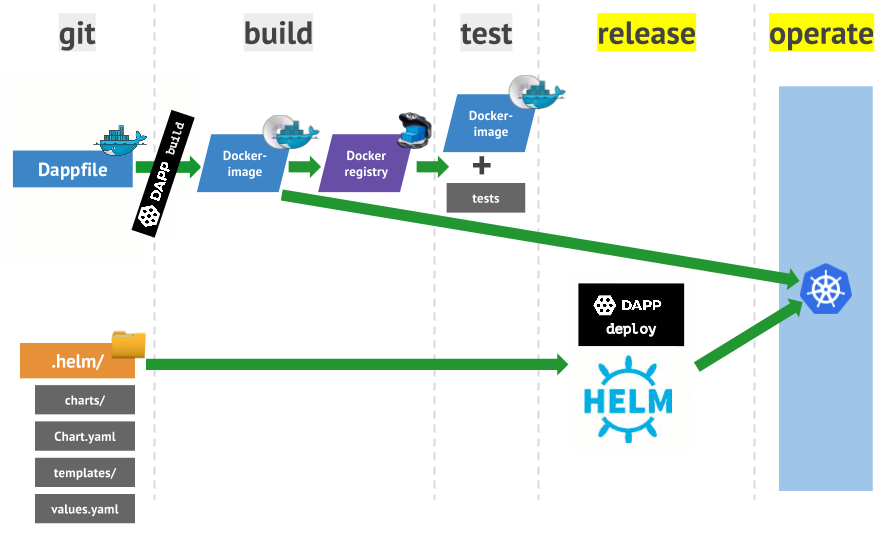
The original goal of Helm was to provide users with a better way to manage all the [Kubernetes](https://azure.microsoft.com/services/kubernetes-service/?WT.mc_id=containers-19838-ludossan) YAML files we create on [Kubernetes](https://azure.microsoft.com/services/kubernetes-service/?WT.mc_id=containers-19838-ludossan) projects.

The path [Helm](https://docs.microsoft.com/azure/aks/kubernetes-helm?WT.mc_id=containers-19838-ludossan) took to solve this issue was to create Helm [Charts](https://docs.microsoft.com/azure/aks/kubernetes-helm?WT.mc_id=containers-19838-ludossan). Each chart is a bundle with one or more [Kubernetes](https://azure.microsoft.com/services/kubernetes-service/?WT.mc_id=containers-19838-ludossan) manifests – a chart can have child charts and dependent charts as well.

This means that Helm installs the whole dependency tree of a project if you run the install command for the top-level chart. You just a single command to install your entire application, instead of listing the files to install via kubectl.

Helm also keeps a release history of all deployed charts, so you can go back to a previous release if something went wrong.

[Helm](https://docs.microsoft.com/azure/aks/kubernetes-helm?WT.mc_id=containers-19838-ludossan) supports [Kubernetes](https://azure.microsoft.com/services/kubernetes-service/?WT.mc_id=containers-19838-ludossan) natively, which means you don't have to write any complex syntax files or anything to start using Helm. Just drop your template files into a new chart and you're good to go.



## Contents of a Helm Chart

Helm charts are packaged directories that, at a minimum, must contain these files:

**Chart.yaml** - A YAML file containing information about the chart

**values.yaml** - The default configuration values for this chart

**templates/** - A directory of templates that, when combined with values, will generate valid Kubernetes manifest files.

**charts/** - A directory containing any charts upon which this chart depends

Packaging and Deployment

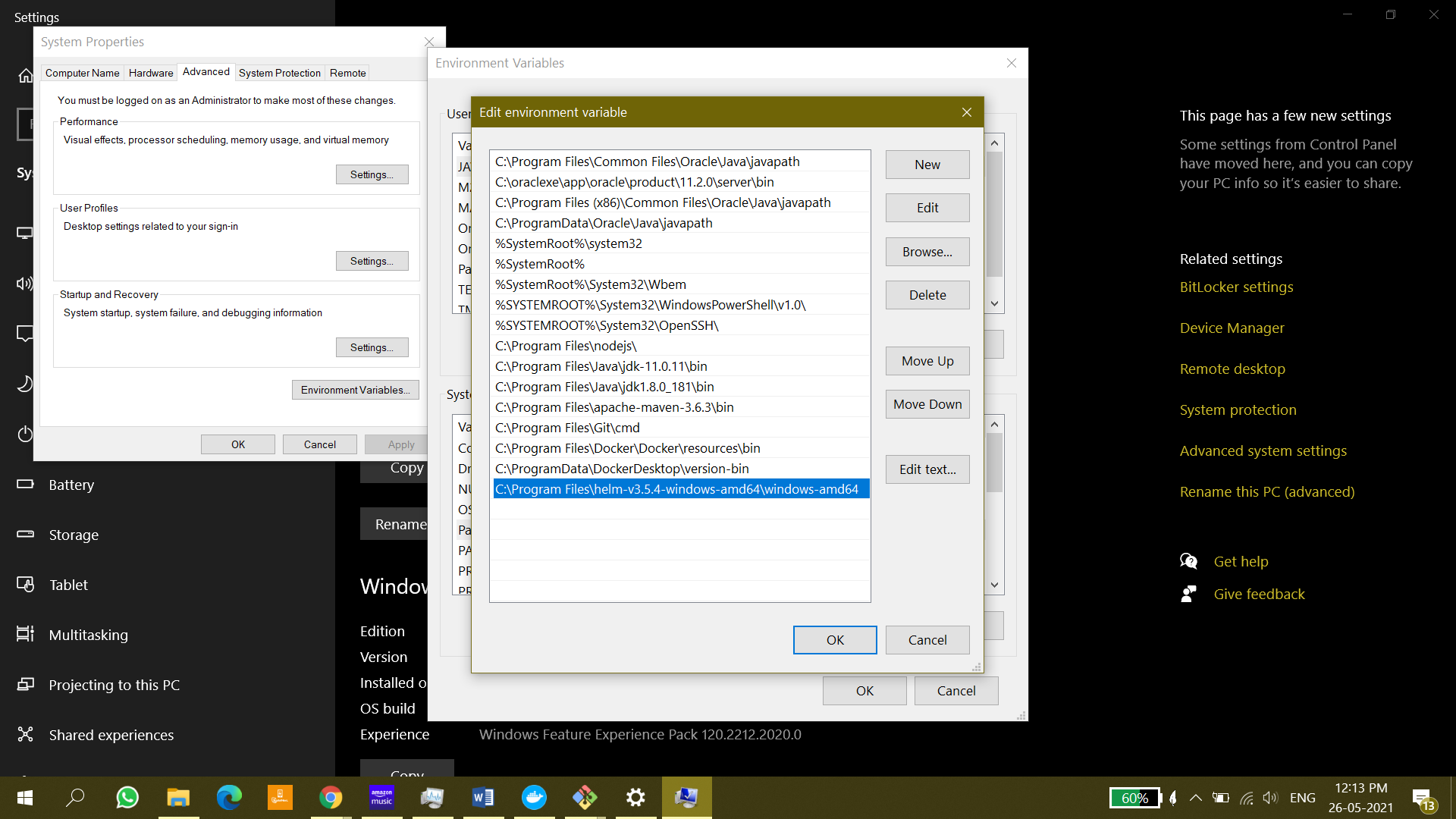
Helm charts are packaged simply by zipping up files into a tarball (.tgz) file

# Install Helm on windows

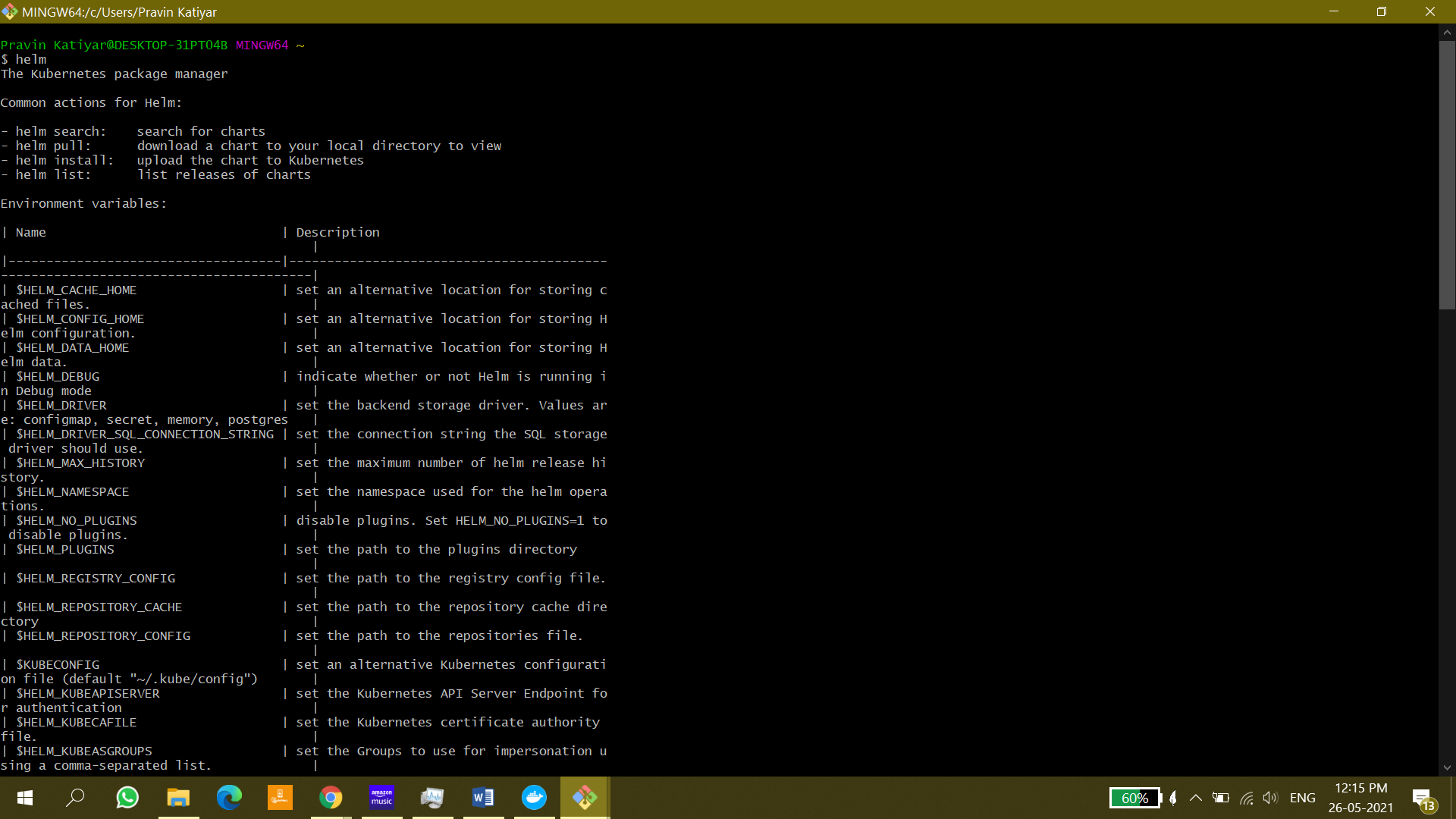
Download helm =>https://get.helm.sh/helm-v3.5.4-windows-amd64.zip

C:\Program Files\helm-v3.5.4-windows-amd64\windows-amd64

Set environment variables

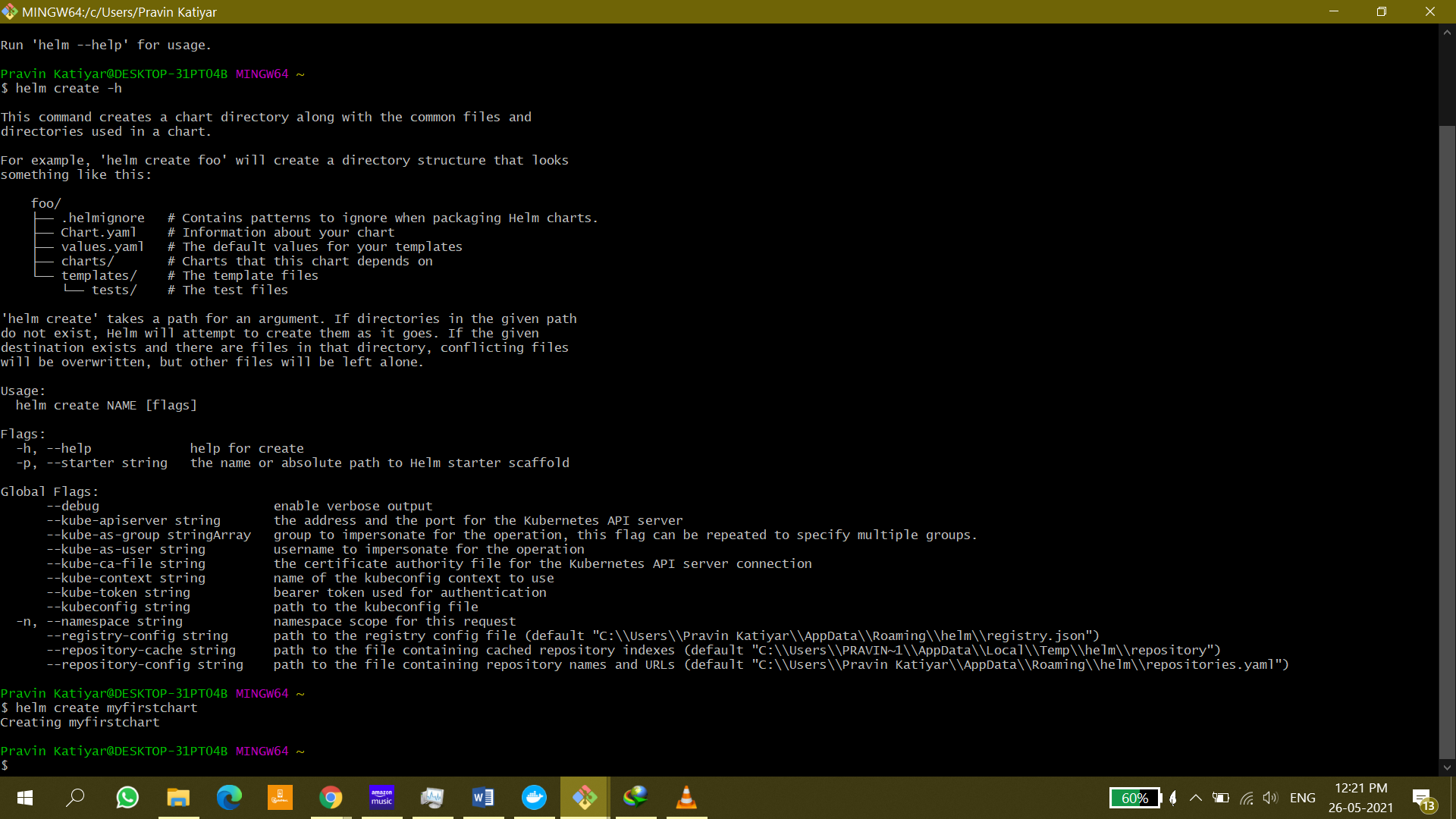


**To check helm is working or not type: “helm” on cmd**

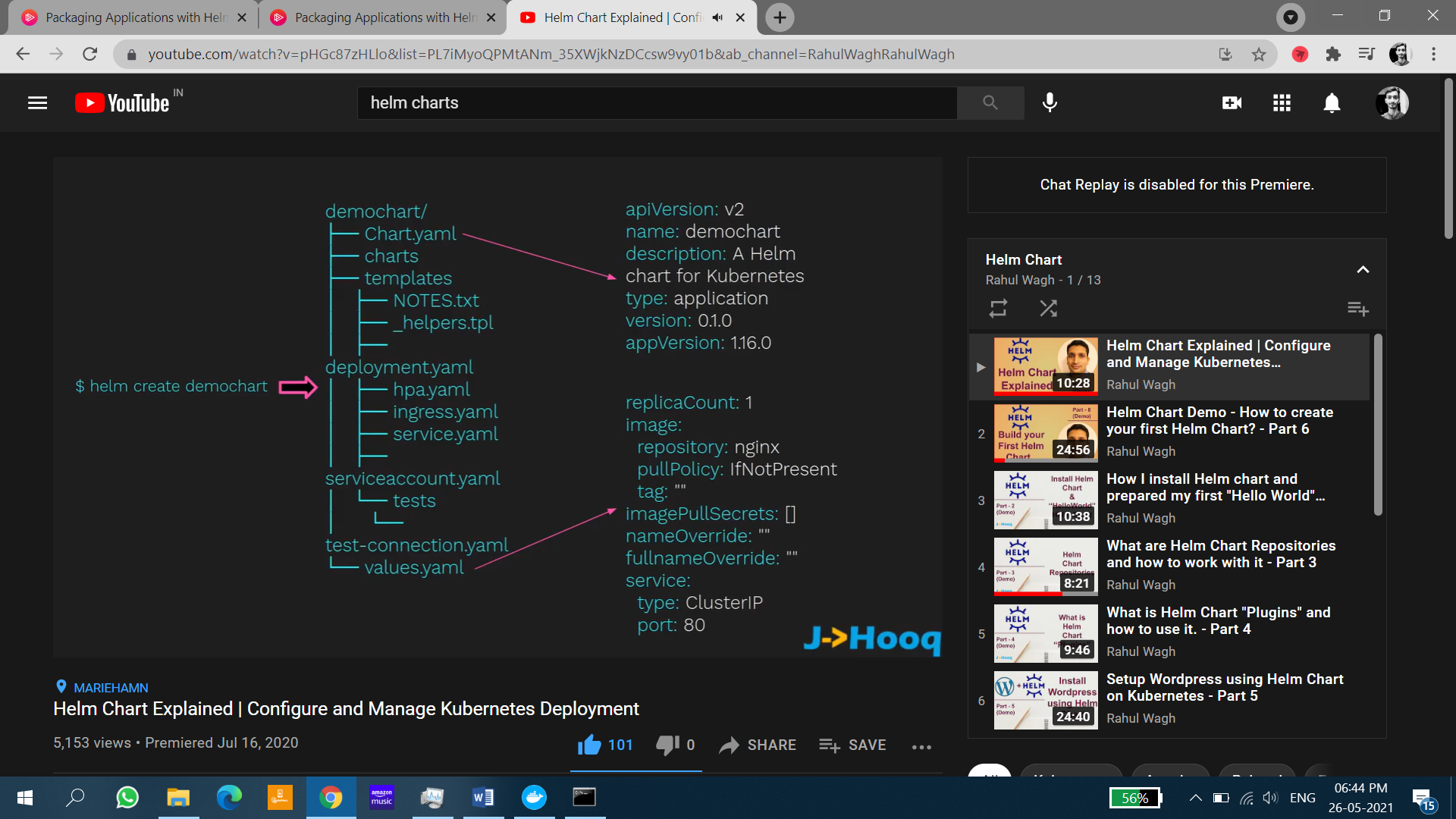


# Creating Helm chart

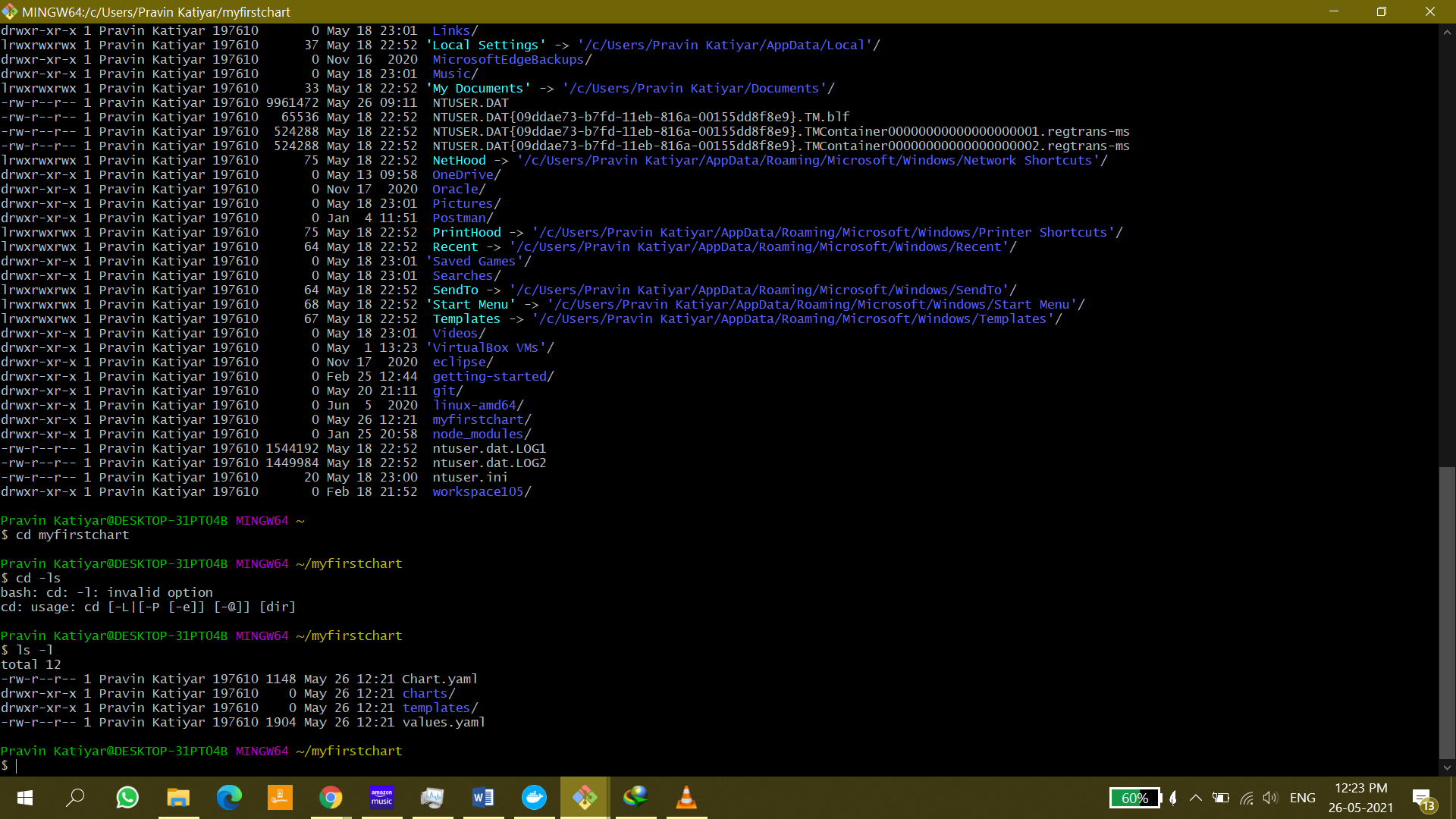
1. Create chart =>helm create [myfirstchart]



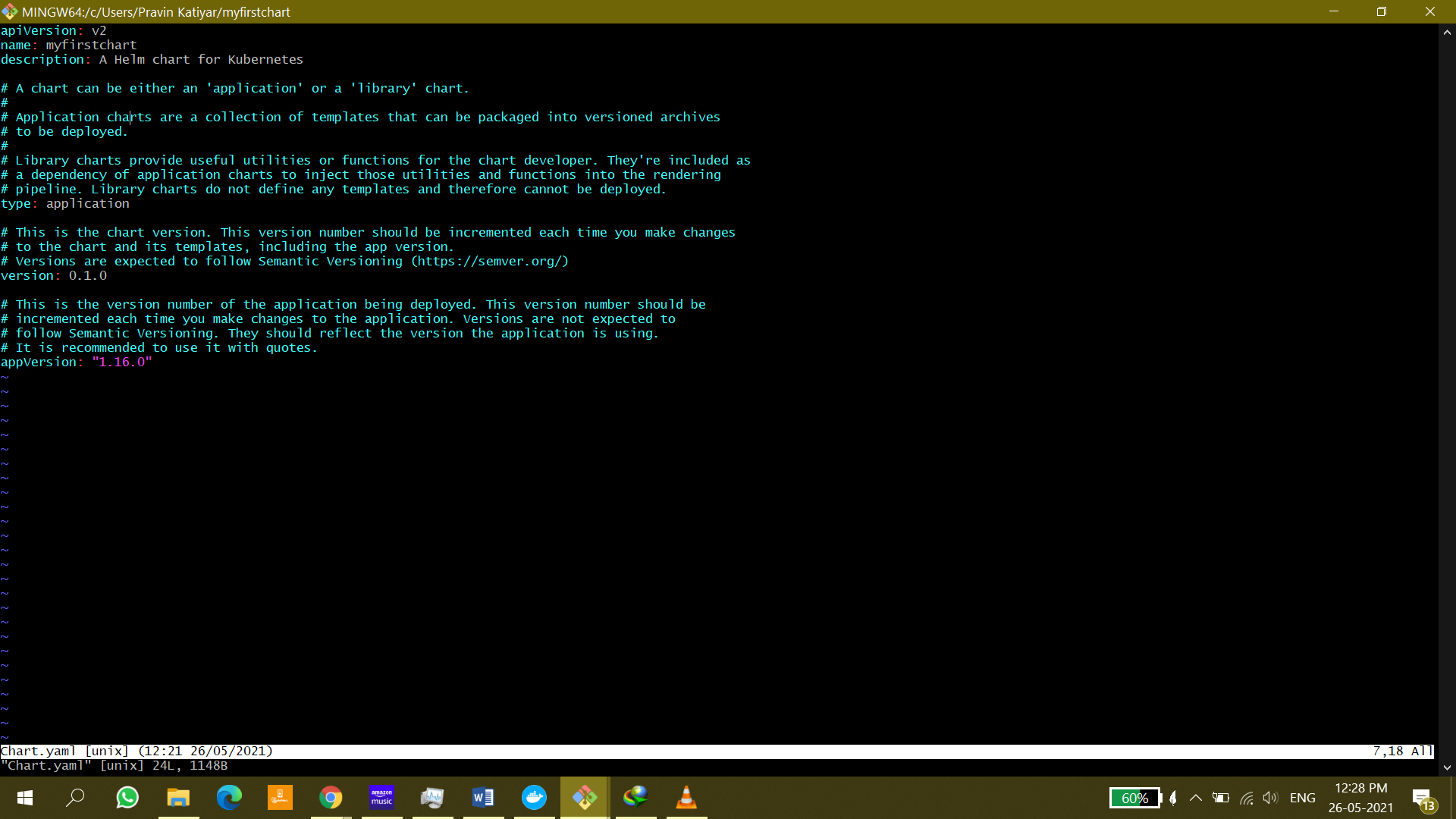
Structure of chart looks like something-



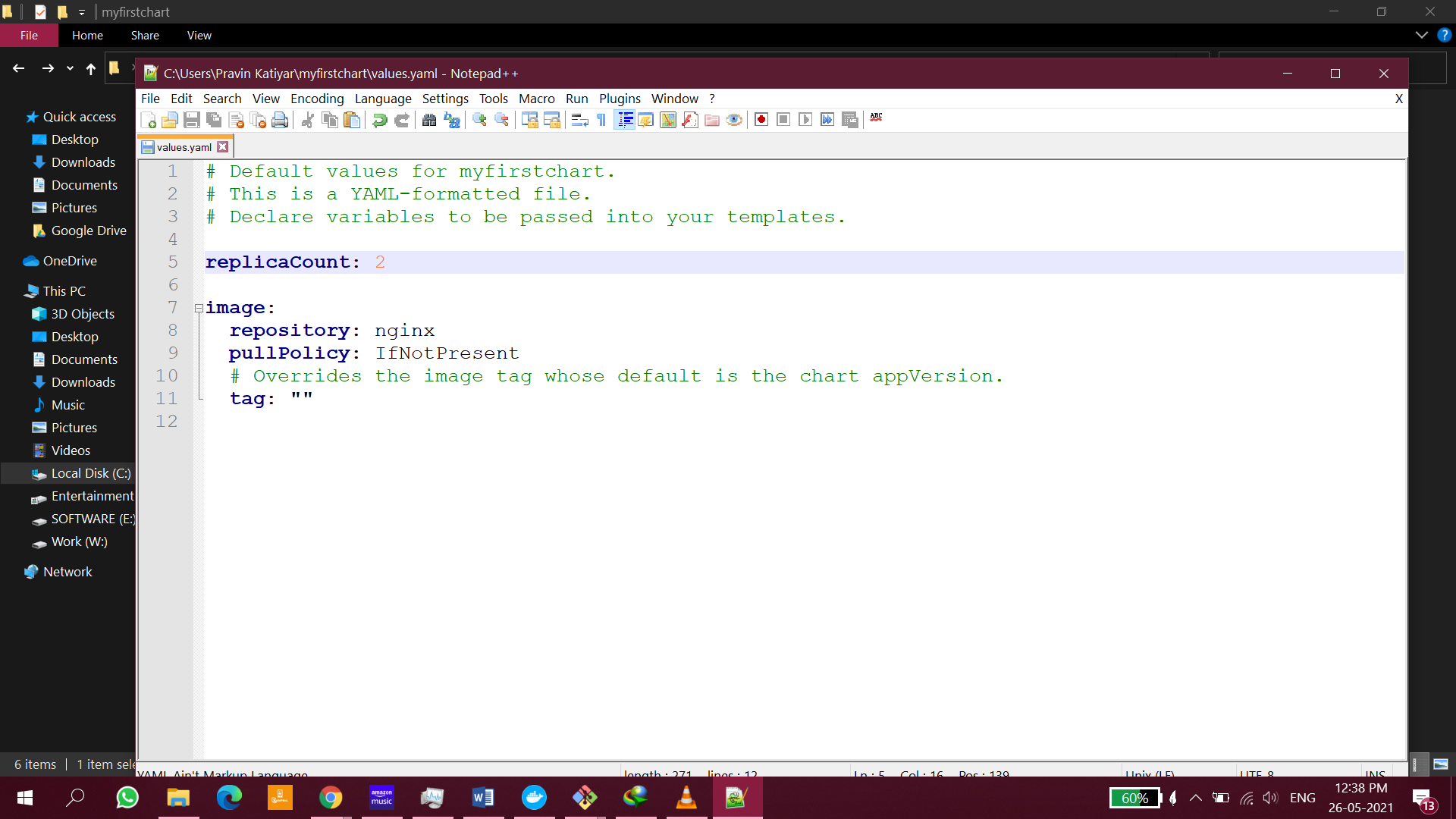
2: view the content of myfirstchart =>ls -l



3: Viewing content of any file =>vi Chart.yaml



4: Edit values.yaml file (remove unnecessary labels to make it simple to learn)



5 Delete all the files inside template folder

**6 Create deployment.yml file inside template folder**

apiVersion: apps/v1

kind: Deployment

metadata:

name: sample-webapp-dep-helm

spec:

replicas: {{ .Values.replicaCount }}

selector:

matchLabels:

app: sampleapp

template:

metadata:

labels:

app: sampleapp

spec:

containers:

- name: sampleapp

image: pravin99/samplewebapp

ports:

- containerPort: 80

**7 .Create service.yml file inside template folder**

apiVersion: v1

kind: Service

metadata:

name: webapp-svc-helm

labels:

app: sampleapp

spec:

type: NodePort

ports:

- port: 80

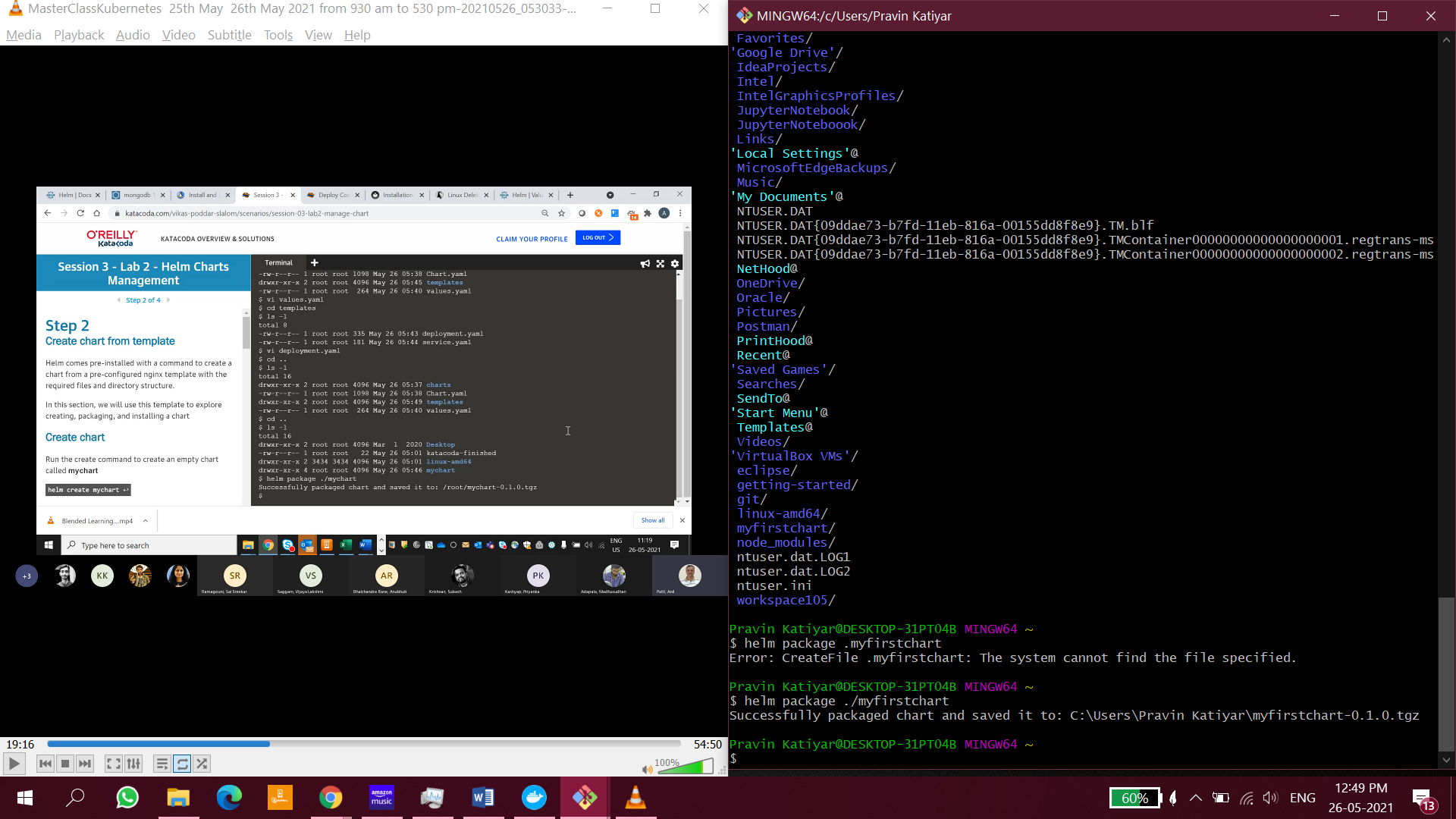
nodePort: 30085

selector:

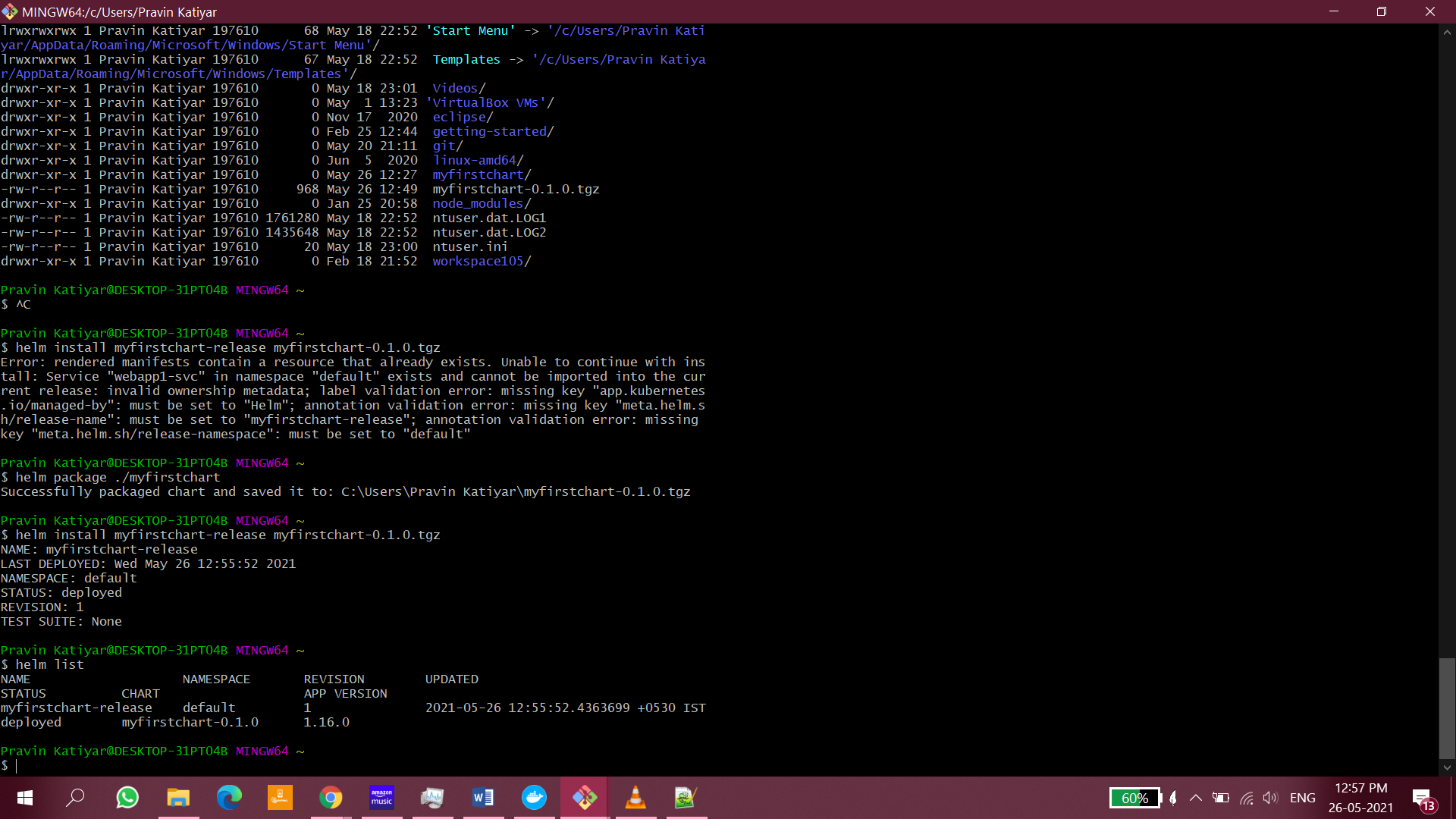
app: sampleapp

**NodePort=>30085**

8. Package the myfirstchart =>helm package ./myfirstchart

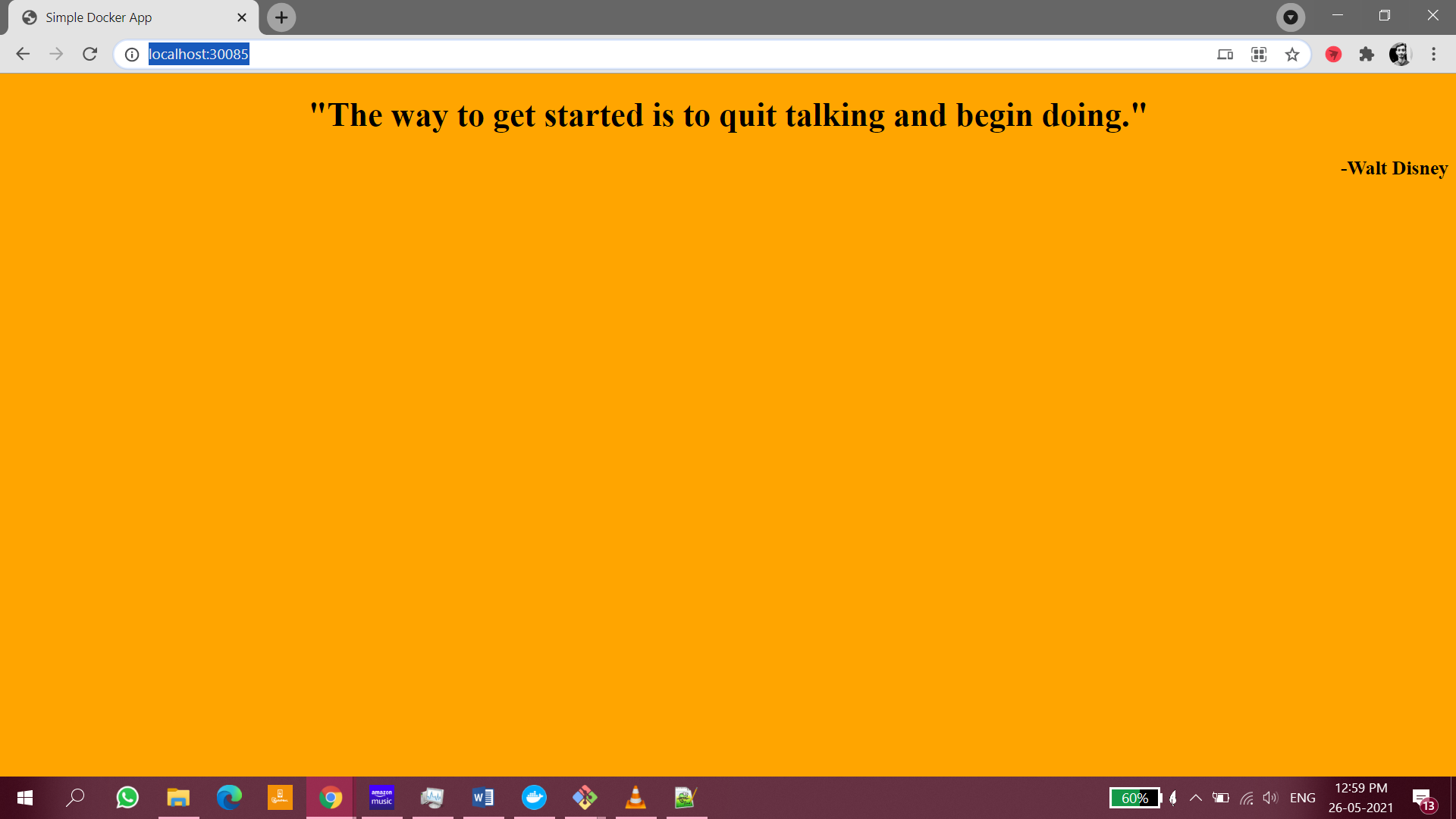


**9.** Create a release of chart

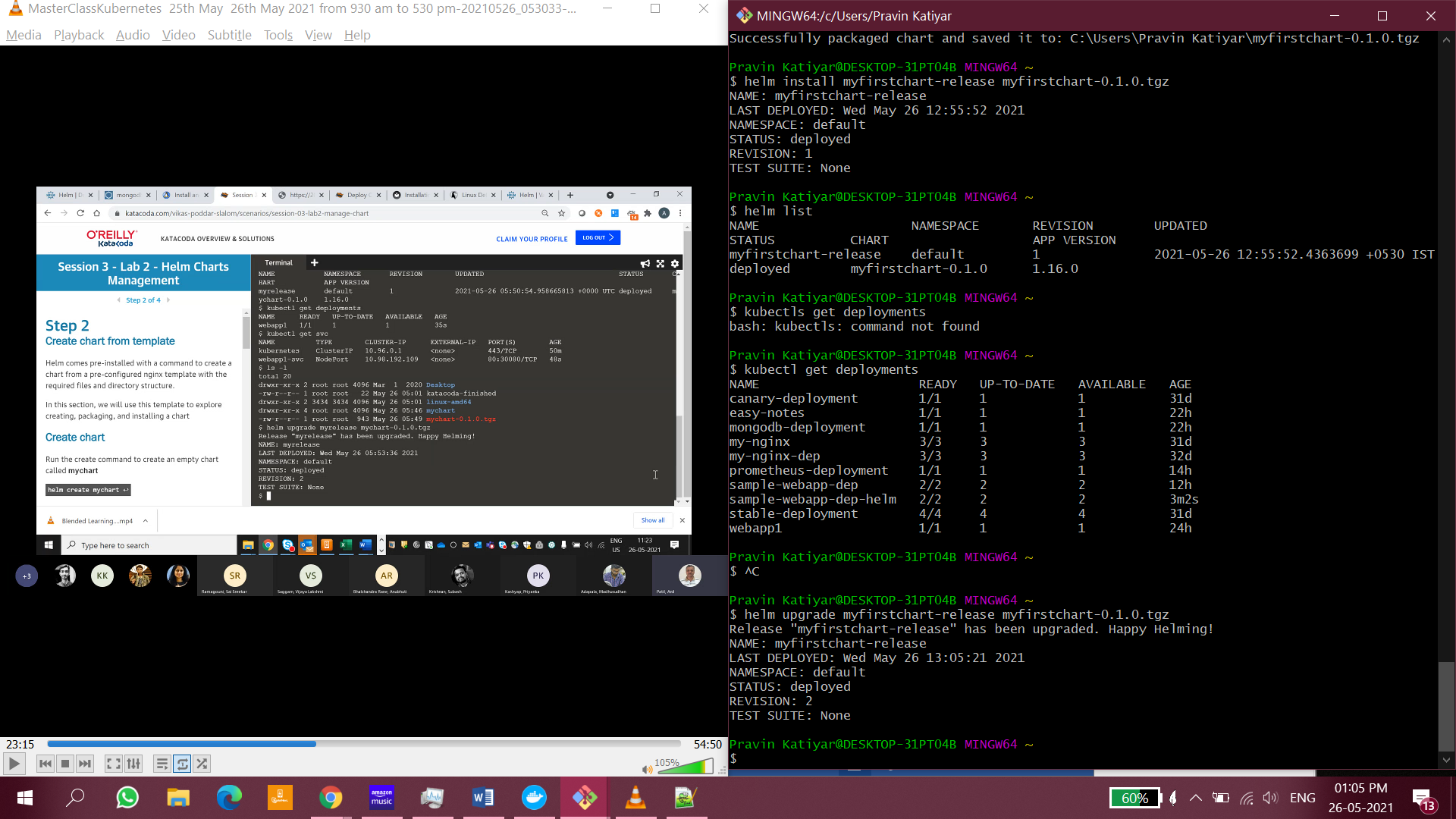


**We can see REVISION: 1**

**10.** Hit localhost: 30085 in the URL

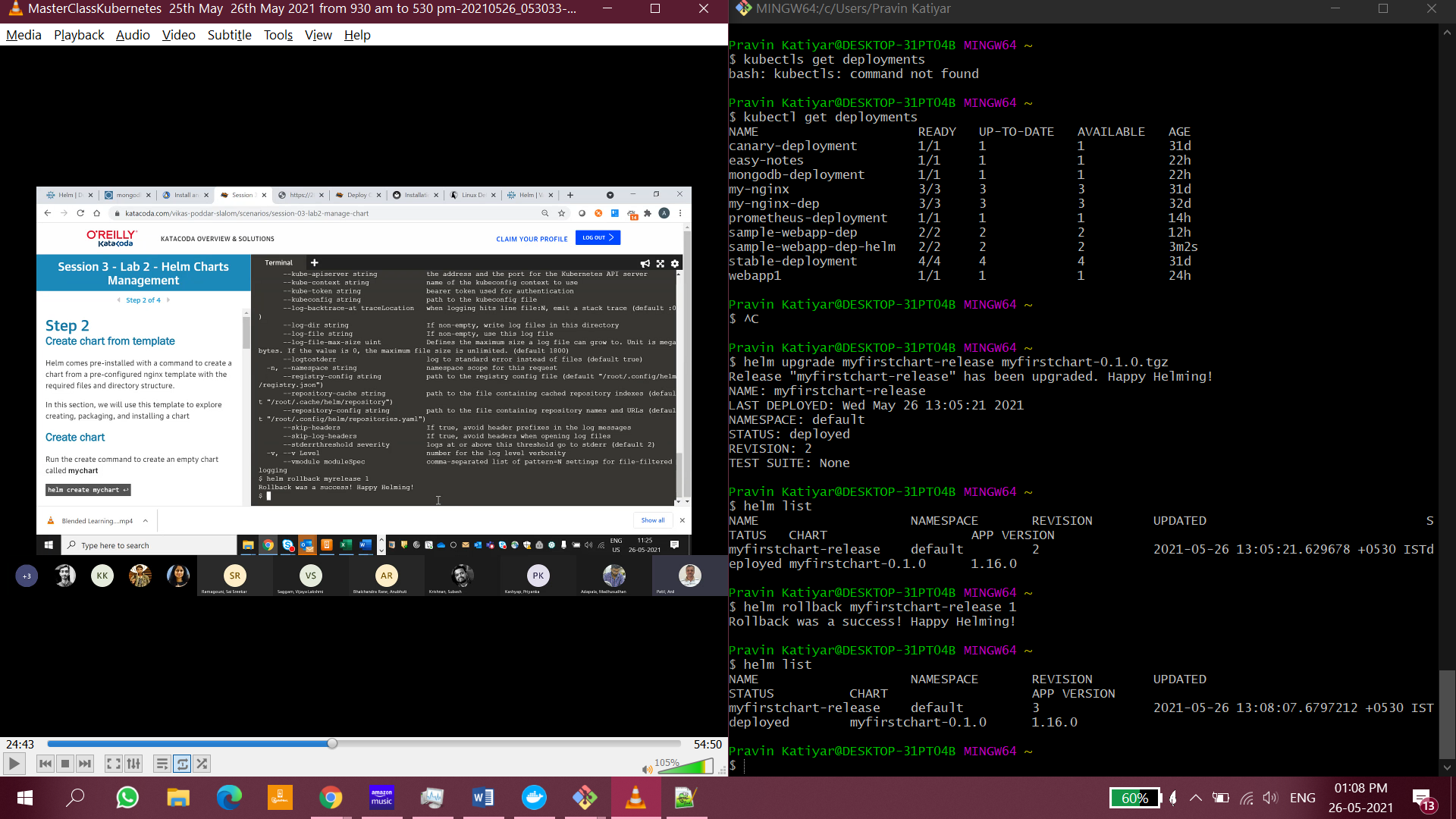


# Upgrading Helm Chart



**We can see REVISION: 2**

# Rollback Helm Chart to earlier versions



# Uploading Helm Chart to registry

refer => https://helm.sh/docs/topics/registries/

