

**Lab Manual- Create Own Helm Chart**

**Prepared for**:

**Date:** 18th Dec 2023

**Prepared by:**

Document Name: Lab Manual **Document Number** AZLabn916

**Contributor:**

Contents

[1. Objective 3](#_Toc156546314)

[2. Create Helm Chart From Scratch 4](#_Toc156546315)

[3. Create Template in Helm Chart 6](#_Toc156546316)

[4. Text Processor in Template in Helm Chart 11](#_Toc156546317)

# Objective

Helm charts are structured like this:

mychart/

Chart.yaml

values.yaml

charts/

templates/

...

The **templates/** directory is for template files. When Tiller evaluates a chart, it will send all of the files in the **templates/** directory through the template rendering engine. Tiller then collects the results of those templates and sends them on to Kubernetes.

The **values.yaml** file is also important to templates. This file contains the *default values* for a chart. These values may be overridden by users during helm install or helm upgrade.

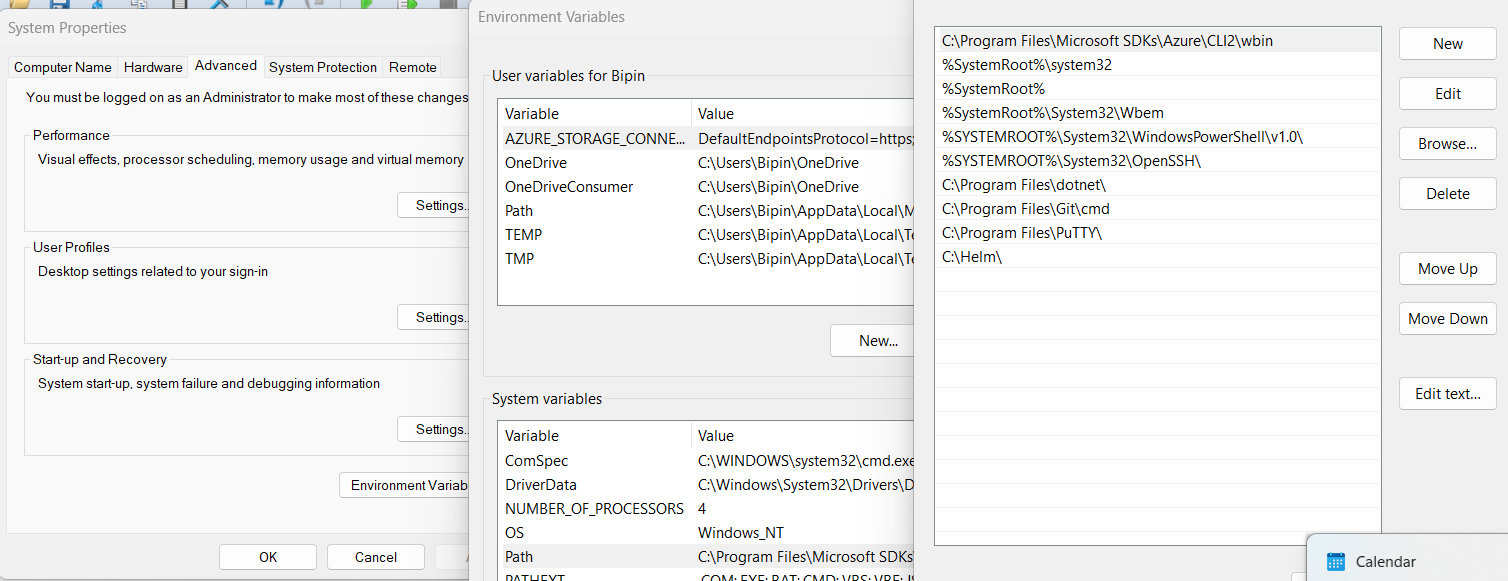
The **Chart.yaml** file contains a description of the chart. You can access it from within a template. The charts/ directory *may* contain other charts (which we call *subcharts*). Later in this guide we will see how those work when it comes to template rendering.

**But another step you can take as you can convert that chart into standard Kubernetes YAML. Therefore, you can apply that to any cluster without having Helm present. It's a kind of optional**

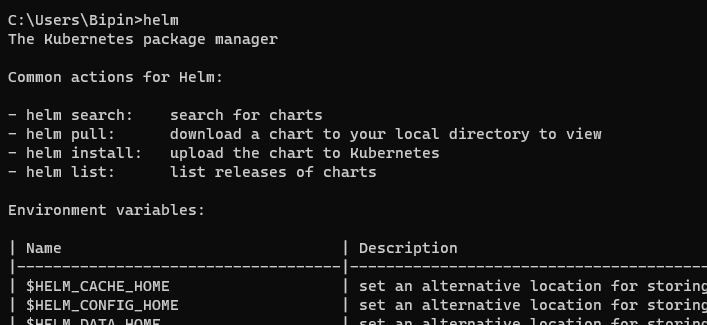
<https://get.helm.sh/helm-canary-windows-amd64.zip>

or

<https://helm.sh/docs/intro/install/>



helm



# Create Helm Chart From Scratch

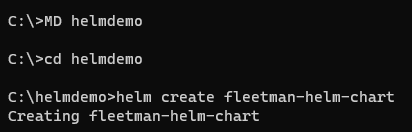
md helmdemo

cd helmdemo



1. Execute the following command to create the chart boilerplate. It creates a chart with the name fleetman-helm-chart with default files and folders.

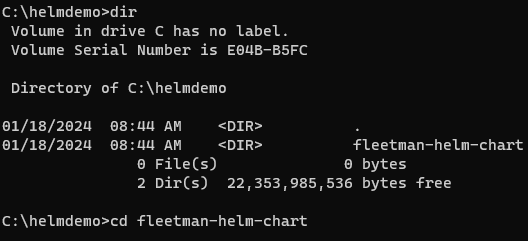
helm create fleetman-helm-chart



1. List the content of the chart

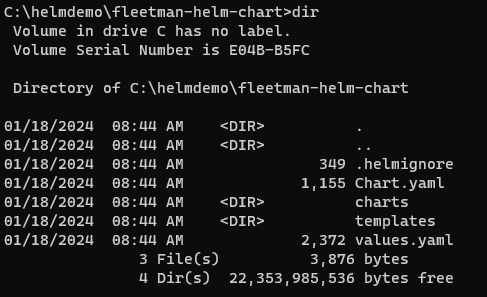
dir

cd fleetman-helm-chart



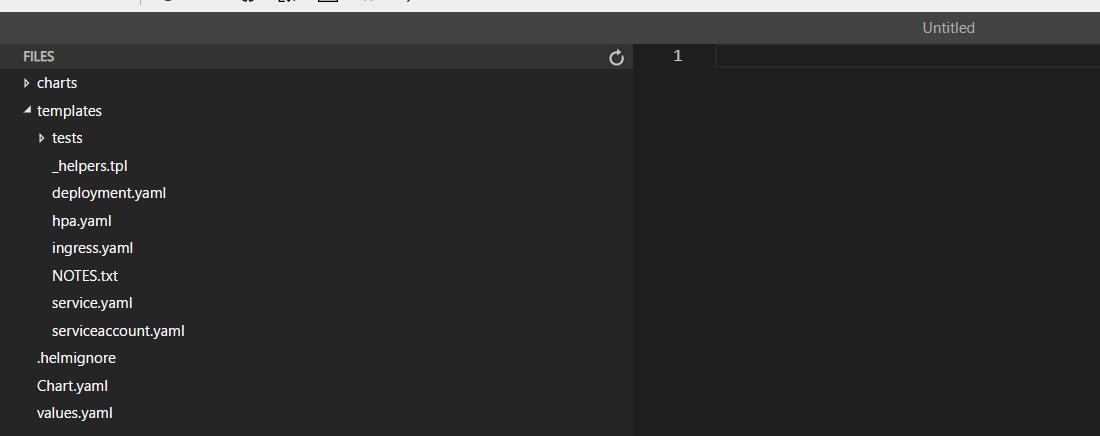
1. List the content of the char

dir



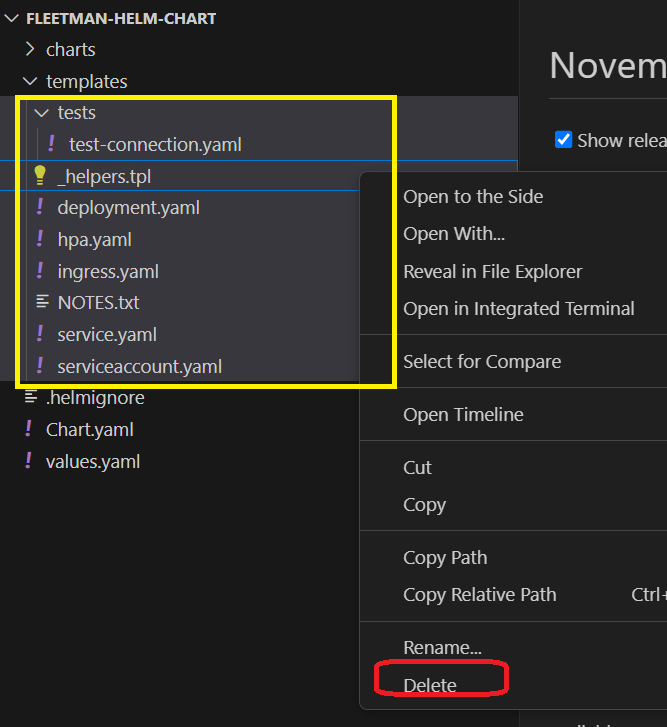
1. Lets open the editor

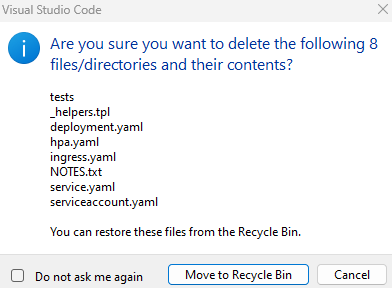
code .

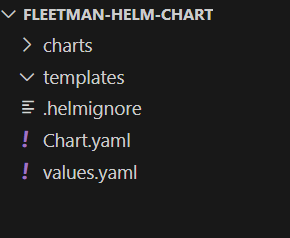


# Create Template in Helm Chart

1. Lets make the **template** folder empty by selecting all Files and Test folder inside template and click delete

****

****

****

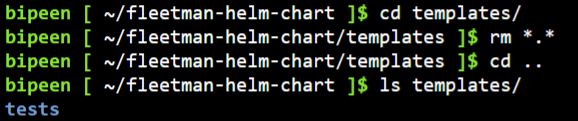
**Optionally if you are in Linux , run below command**

cd templates/

rm \*.\*

cd ..

rm -rf templates/tests/

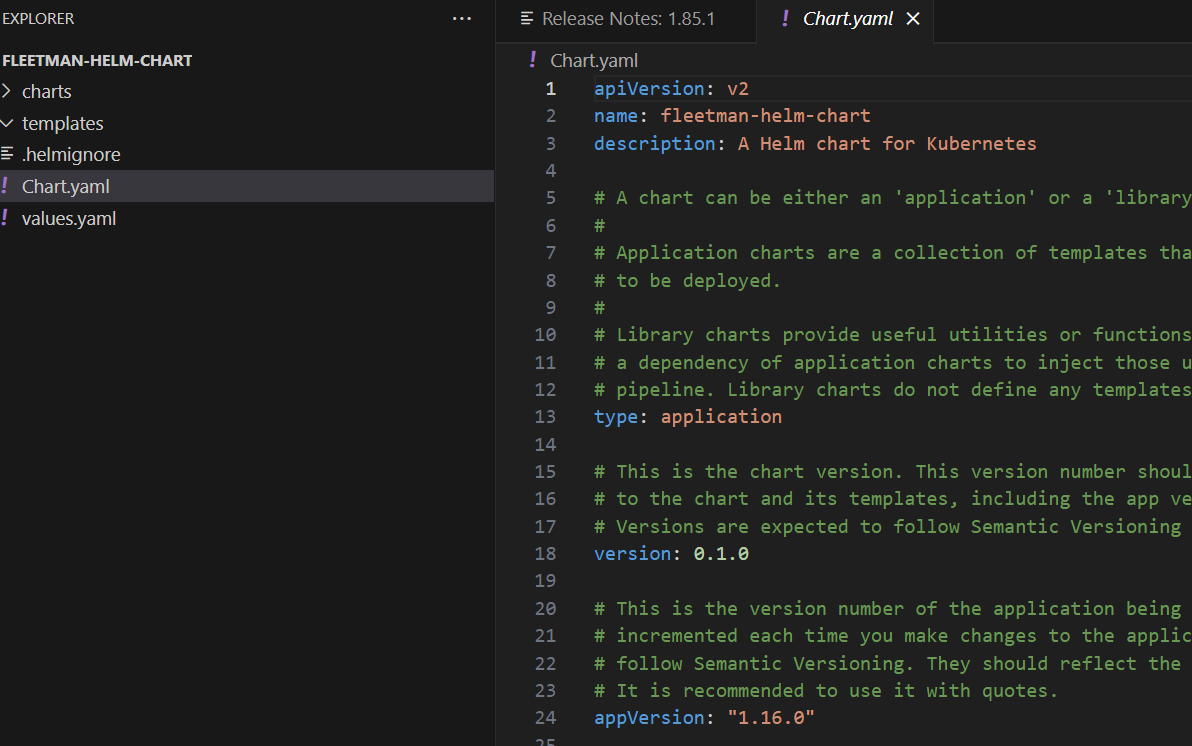




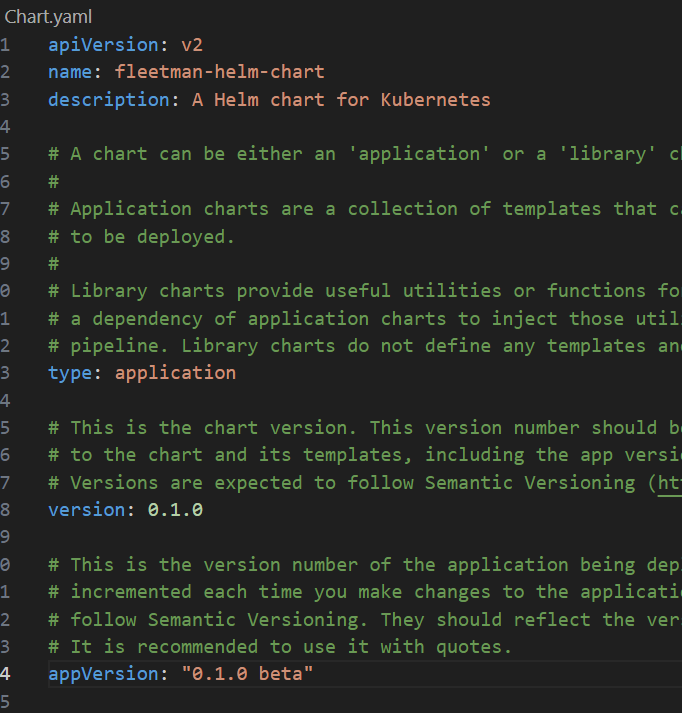
1. Now Open the **Chart.yaml** and note the **Application version** and **appVersion.**

first version is intended to be the version of the charts.Whereas the second version, the app version is the version number of the application itself. Note that the **appVersion** field is not related to the version field. It is a way of specifying the version of the application

**Chart Versioning should be follow Symantec versioning means** Majorversion . Minorversion.Patchversion



1. Now lets change the version of Appversion also to 0.1.0 beta and save the file. Here you note we do not need to follow the Symantec Version. Note that the **appVersion** field is not related to the version field. It is a way of specifying the version of the application

****

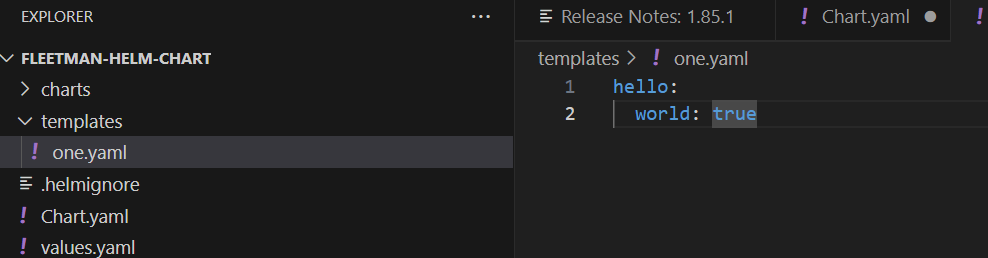
1. Now click the folder called templates. In this folder any yaml file

inside this template folder is going to be sent through a textprocessor, which is going to ultimately generate the **Kubernetes yaml** that we need

Now Lets create a new Yaml file inside template called one.yaml and write very simple valid yamls

hello:

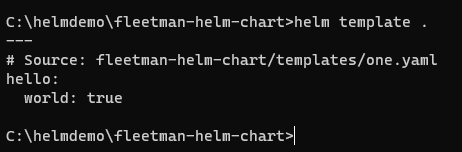
world: true



1. we are in the root directory of the charts. So Helm template dots is going to run

the processor. And can you see that we have some output which is pretty much it's really just echoing the sample YAML that we put in the file. It's add a little bit extra it has put a **document separator**  **---** there. And this is really to help future debugging. It's giving us a link to the source of where that data came from. And that was our one .YAML file

helm template .



1. But you can possibly seen now we can add in now as well. Many files as we like into this Templates folder, let's try it, we'll add a new file called to **two.yaml**

something:

  not:

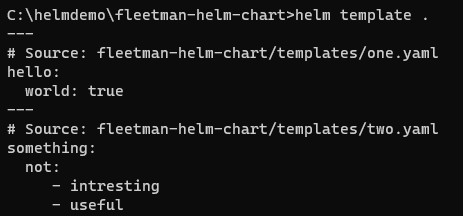
     - intresting

     - useful

1. run the processing, again you can see now that it's taking the entire contents

of that directory, and combining together into a single output.

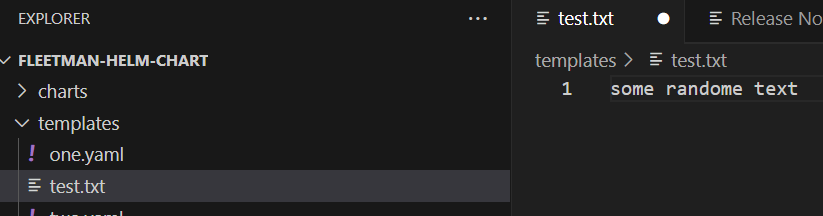
helm template .



1. important to note that, barring a few exceptions, the contents of the templates folder does need to be yaml files, I can verify this by if I put in a new file here called **test.txt**, for example, and just put some random text in there.

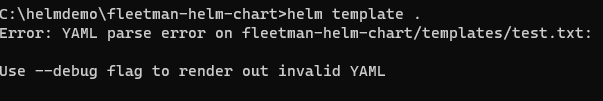
test.txt

some random text

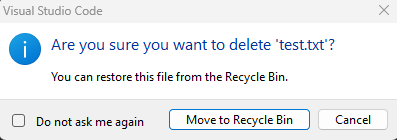


1. Running the template process again, this time you will be getting a **YAML error** on that file, even though it's clearly not a YAML file. So for the most part, this processing, whatever processing is going on here, is expecting YAML files. *There is a little exception to that* , will discuss in later part.

helm template .



1. Lets delete this test file now.



Note : we can combine multiple YAML files together. And it's going to generate a

single output

# Text Processor in Template in Helm Chart

The Helm client and library is written in the Go programming language. The library uses the Kubernetes client library to communicate with Kubernetes.

<https://pkg.go.dev/text/template>

The input text for a template is UTF-8-encoded text in any format. "Actions"--data evaluations or control structures--are delimited by "{{" and "}}"; all text outside actions is copied to the output unchanged.

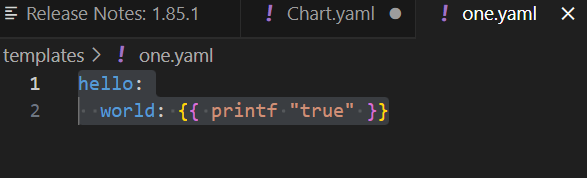
Once parsed, a template may be executed safely

implement some pretty useful Helm charts. What we can do in any of these yaml files is we can use that double curly bracket syntax ourselves, right here. And when the helm processing happens, whatever's inside the curly brackets will be evaluated. Let's try it out here and our one.yaml

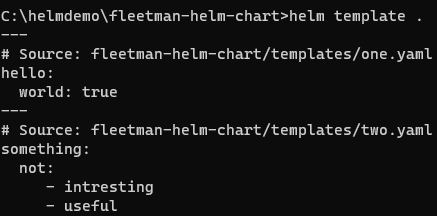
file

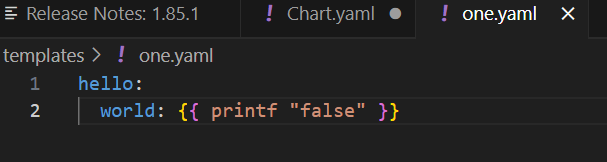
hello:

  world: {{ printf "true" }}

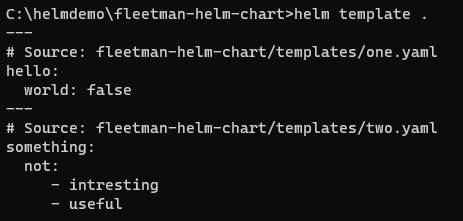


helm template .





helm template .



<https://raw.githubusercontent.com/DickChesterwood/k8s-fleetman/master/_course_files/Intel_AMD_Edition/Going%20Further%20with%20Kubernetes/Helm/fleetman-helm-demo-start.yaml>

apiVersion: apps/v1

kind: Deployment

metadata:

name: webapp

spec:

selector:

matchLabels:

app: webapp

replicas: 1

template: # template for the pods

metadata:

labels:

app: webapp

spec:

containers:

- name: webapp

# Note to deployer - add -dev at the end of here for development version

image: richardchesterwood/k8s-fleetman-helm-demo:v1.0.0

---

apiVersion: v1

kind: Service

metadata:

name: fleetman-webapp

spec:

selector:

app: webapp

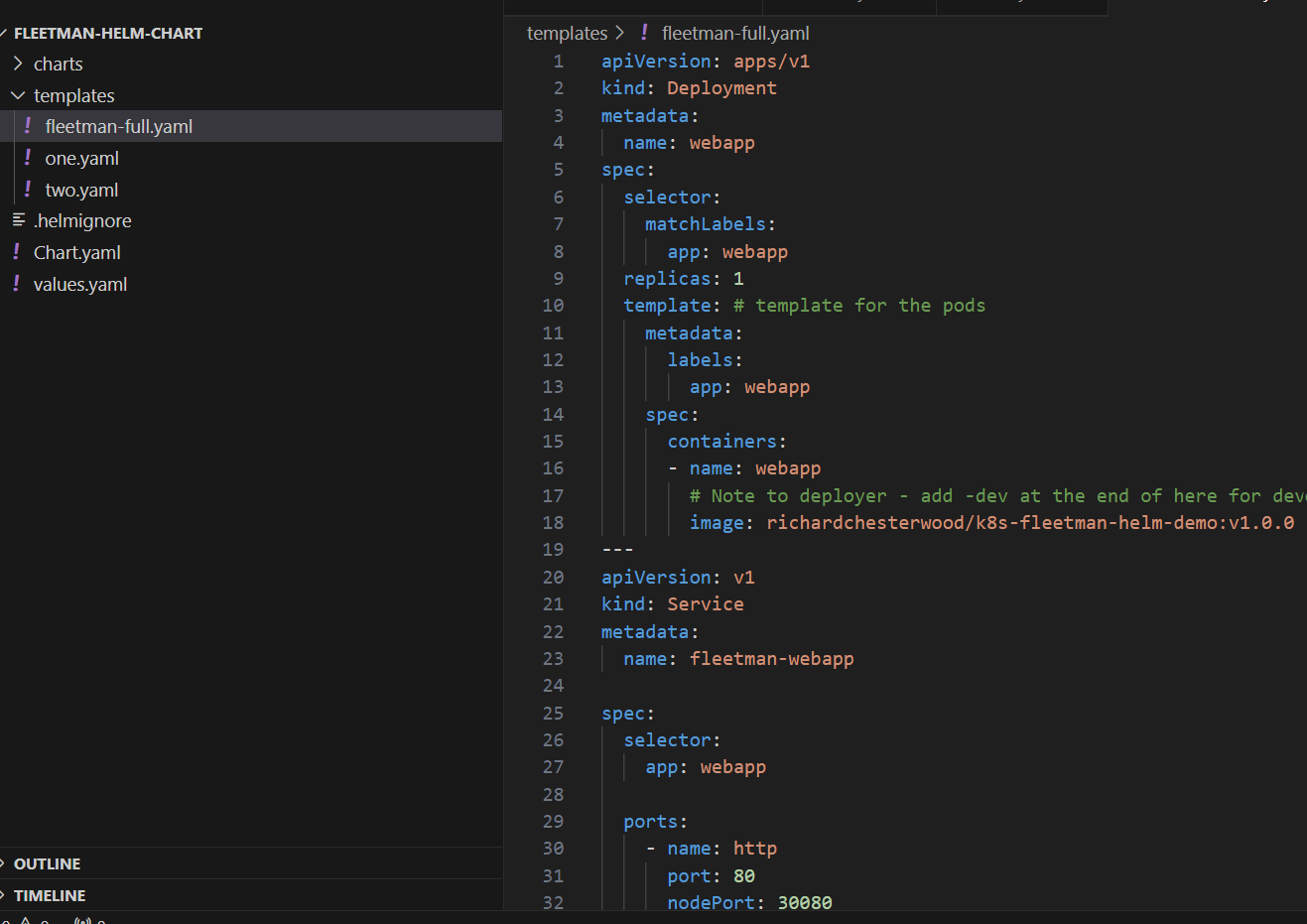
ports:

- name: http

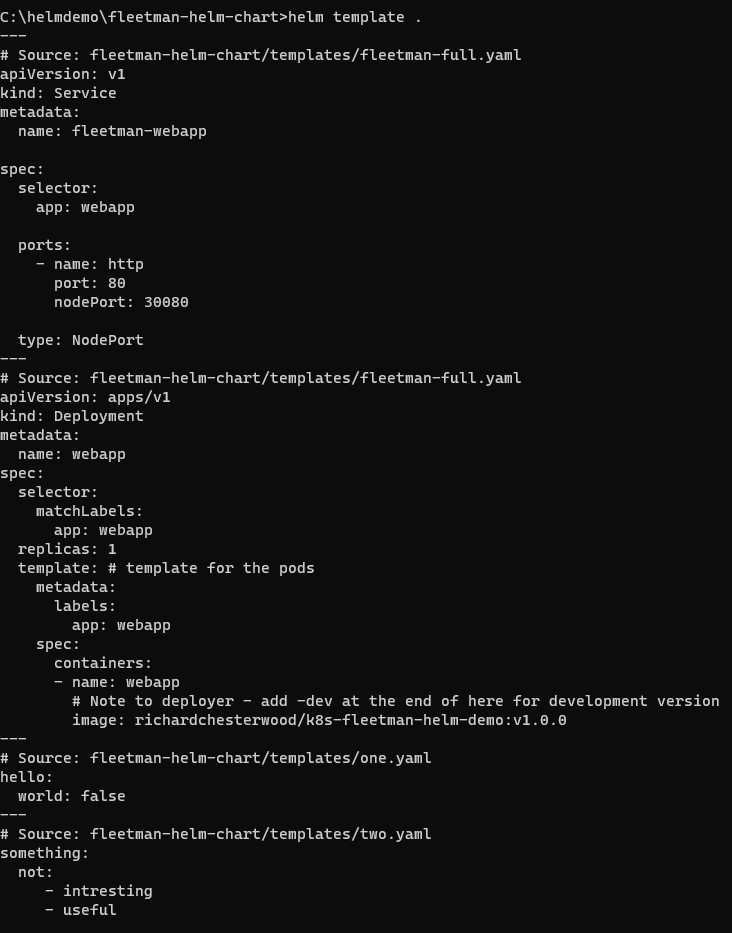
port: 80

nodePort: 30080

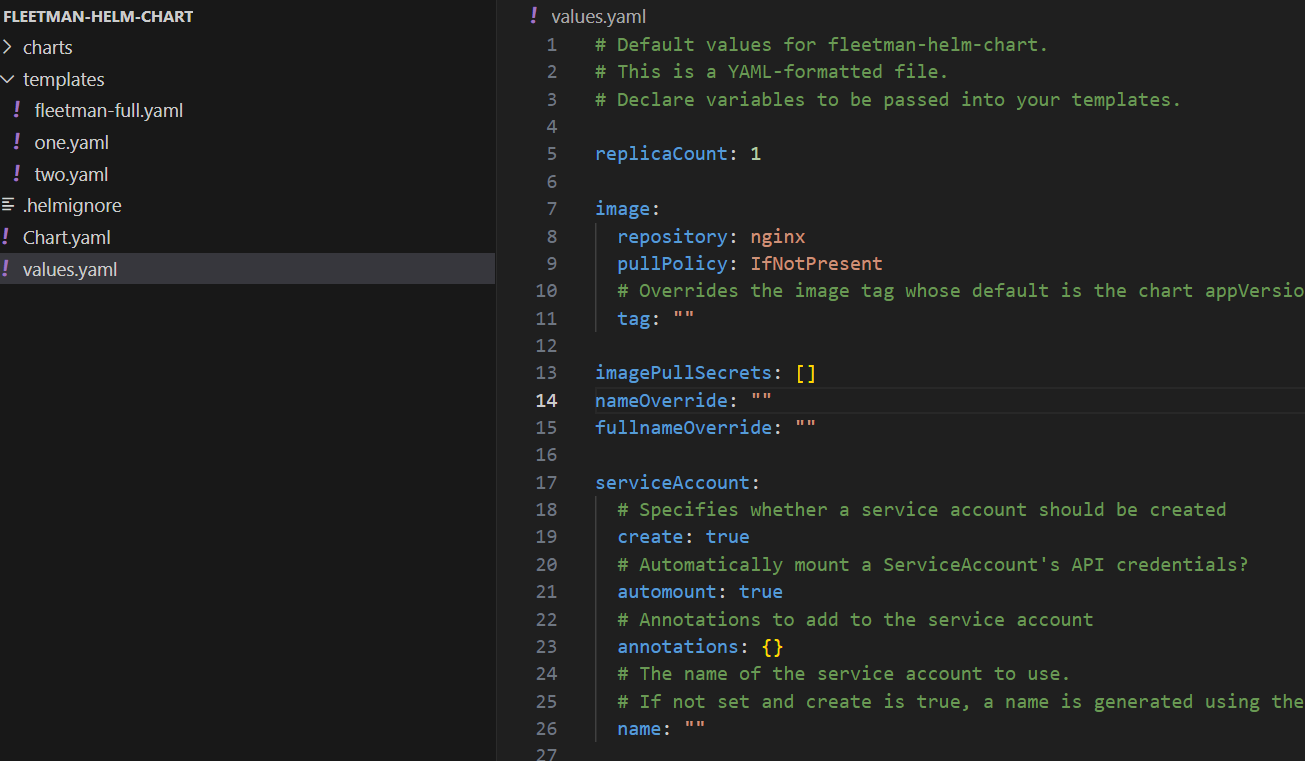
type: NodePort

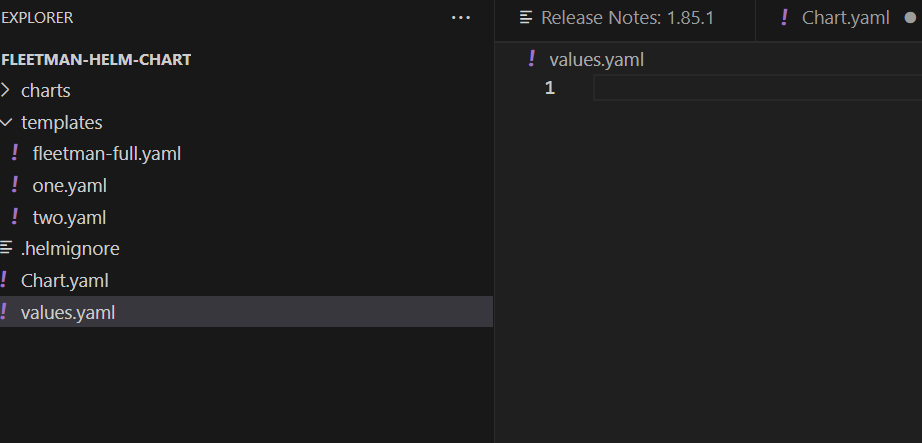


helm template .



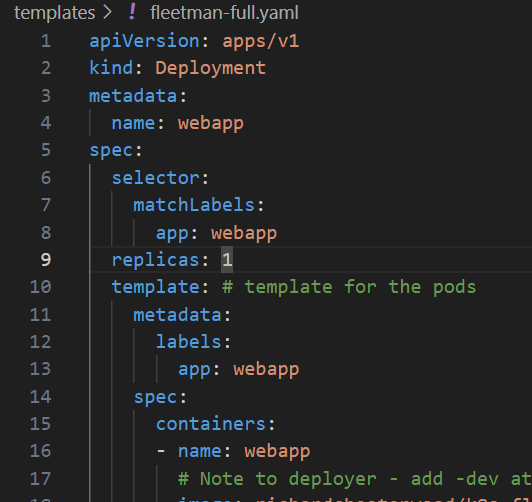
Now click value.yaml and delete everything



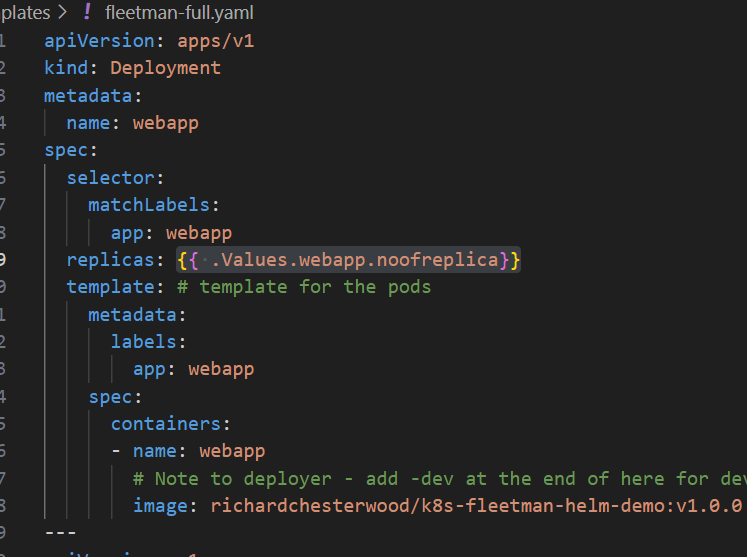


webapp:

  noofreplica: 4



{{ .Values.webapp.noofreplica}}



helm template .

