

**Lab Manual- Helm Function and Pipeline**

**Prepared for**:

**Date:** 18th Dec 2023

**Prepared by:**

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

**Contributor:**

Contents

[1. Objective 3](#_Toc156552796)

[2. Text Processor ( Continue from previous lab) 3](#_Toc156552797)

[3. Helm Functions ( Lower & Upper) 8](#_Toc156552798)

[4. Helm Functions (Default) 11](#_Toc156552799)

[5. Helm Pipeline 14](#_Toc156552800)

[6. Helm Pipeline Chaining 15](#_Toc156552801)

# Objective

Helm has over 60 available functions. Some of them are defined by the Go template language itself. Template functions allow developers to create dynamic and flexible templates for their Kubernetes deployments

Pipelines are a powerful tool for connecting multiple functions together to perform chained tasks. **Pipes allow the output of one function to be used as the input for another function, making it possible to create seamless workflows.**

To create a pipeline, you use the vertical bar character (|) to connect two or more functions together. Pipelines in Helm charts are pretty similar to pipelines you might have seen or used in a Linux environment’s command line.

For example, to print abcd on the screen, we would use a simple command:

echo "abcd"

But if we want to take the output of this command and then process it in some way, we can use a pipeline in another command. In this case, we take the normal output of the **echo** command - which would be **abcd** - pass it to the **tr** command, which transforms all letters to uppercase, and then outputs the final result, as shown below:

user@debian:~$ echo "abcd" | tr a-z A-Z

ABCD

# Text Processor ( Continue from previous lab)

**PART 1 (Background from Previous lab):**

In continuation from last lab where we have created on yaml in Template directory (summary as below)

* Copy the Code from github URL

<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:

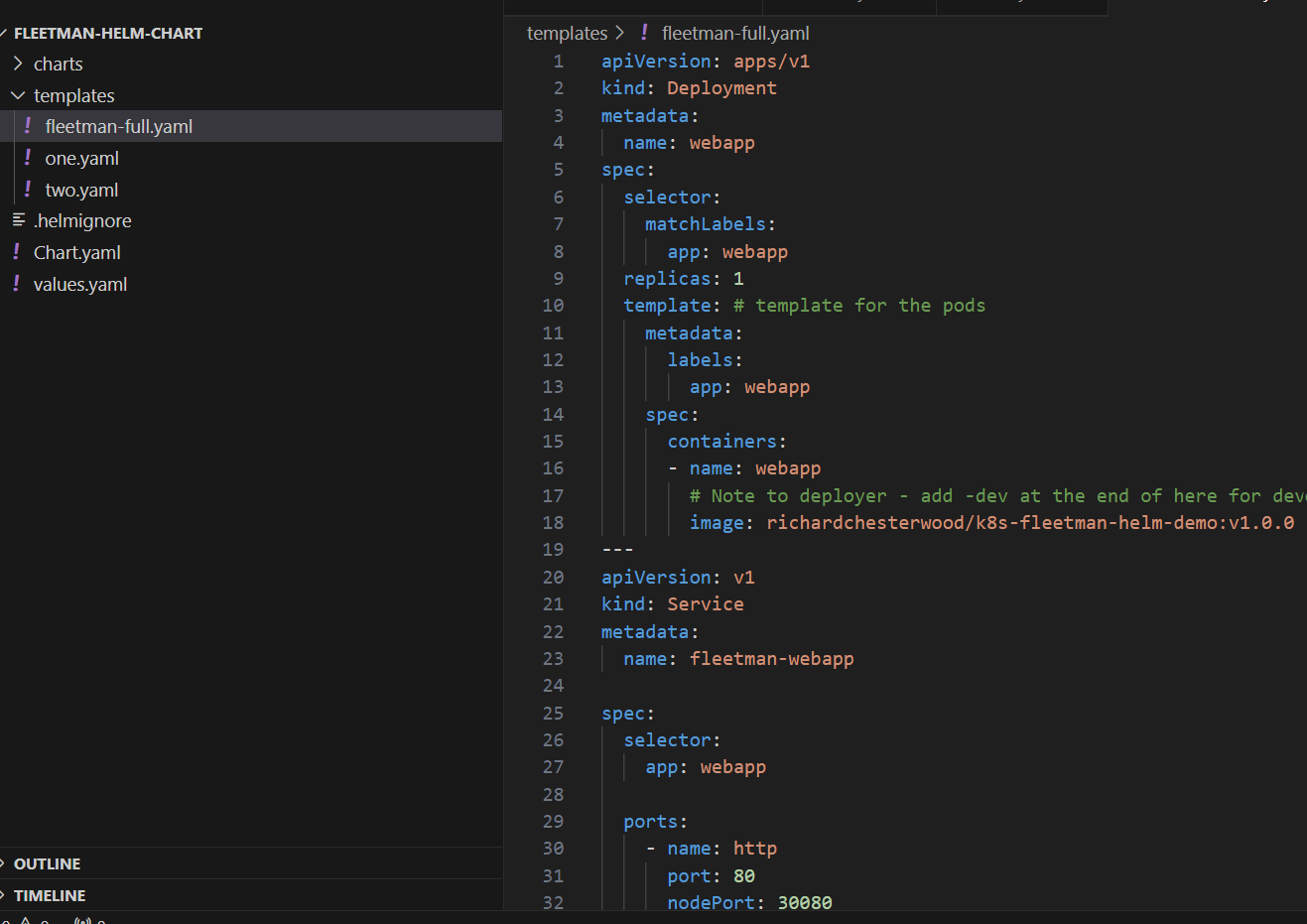
- name: http

port: 80

nodePort: 30080

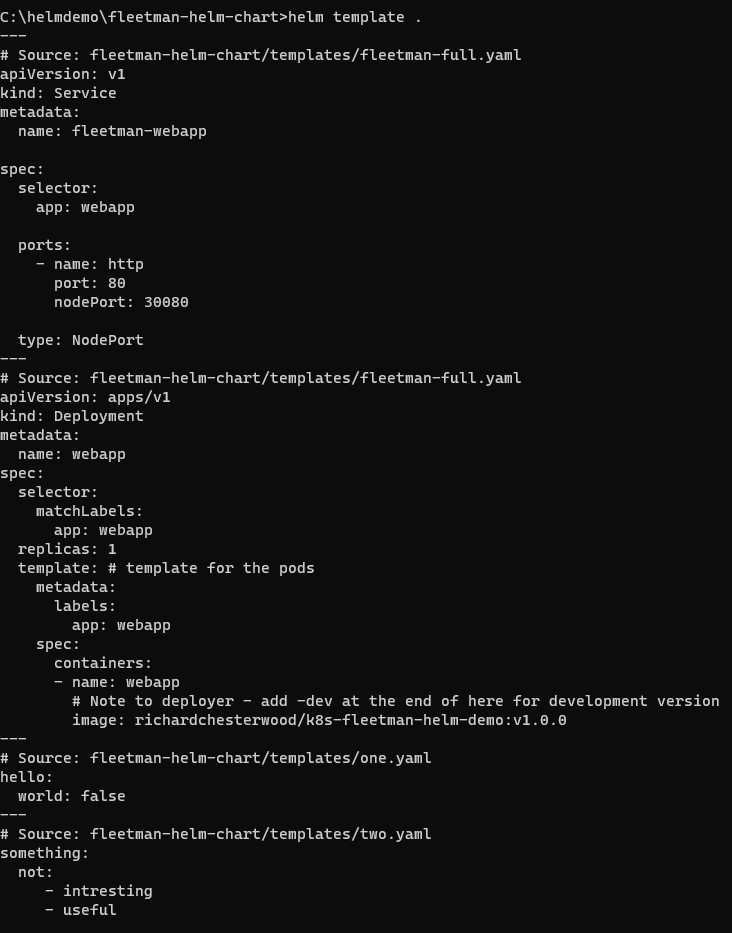
type: NodePort

* Paste the code in new File under Template folder

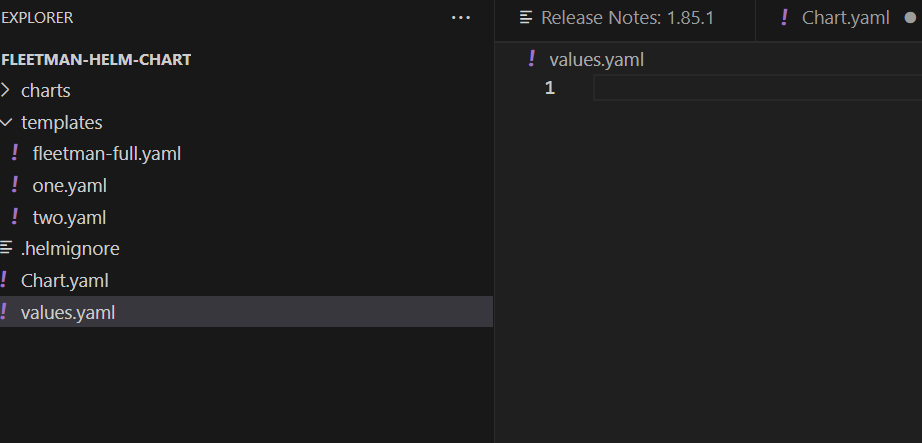


* Run the processing to confirm if everything is ok

helm template .



* Now click value.yaml and delete everything



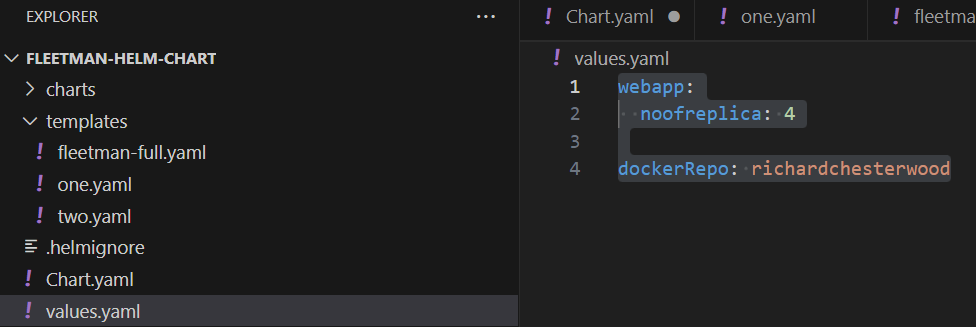
**PART 2**

* Now, let's consider an example in the YAML file where we want to dynamically use the Docker repository name. For that, we can navigate to the values.yaml file and enter the Docker repository name as " **richardchesterwood**."

webapp:

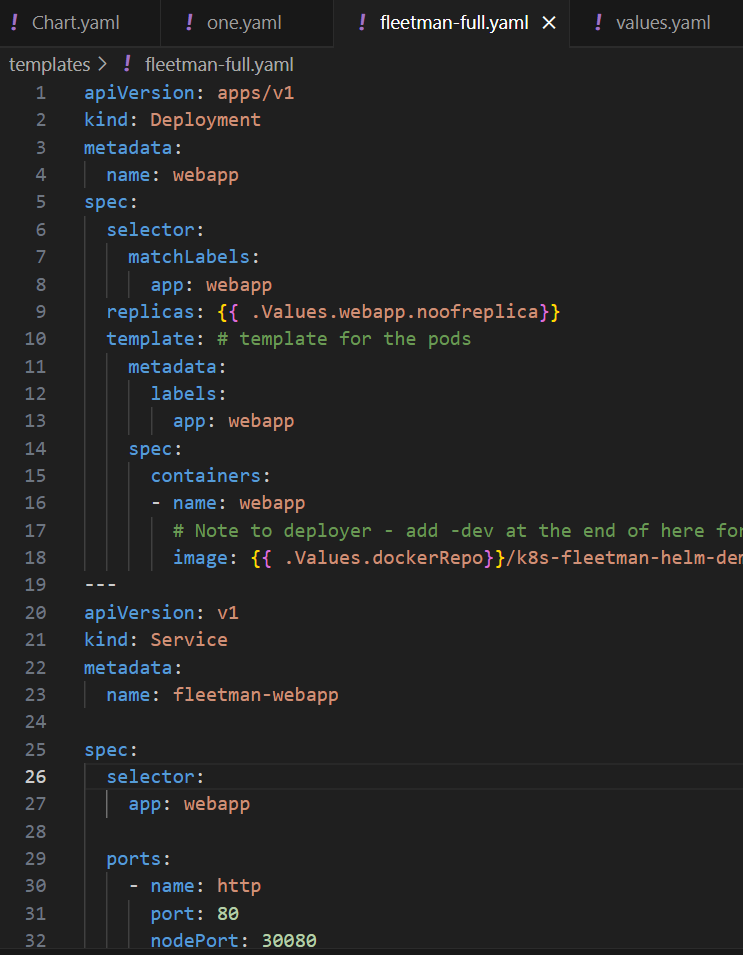
  noofreplica: 4

dockerRepo: "richardchesterwood"



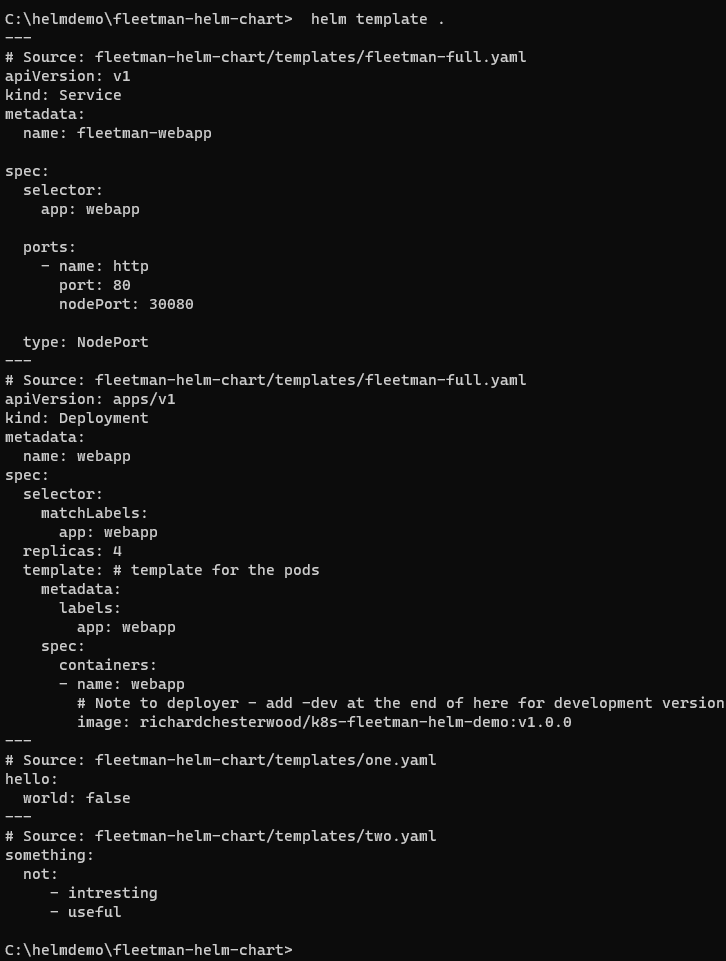
* Now we can go to **Fleetman-full.yaml** and change the value to Image as below

image: {{ .Values.dockerRepo}}/k8s-fleetman-helm-demo:v1.0.0



* Now Run the processing again

helm template .



# Helm Functions ( Lower & Upper)

You can have full list of helm function as below

<https://helm.sh/docs/chart_template_guide/function_list/>

We will use string function in this lab

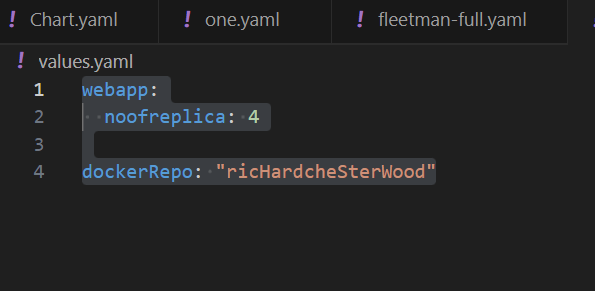


* Lets assume that User has put Docker Repository name in wrong case into the value.yaml

webapp:

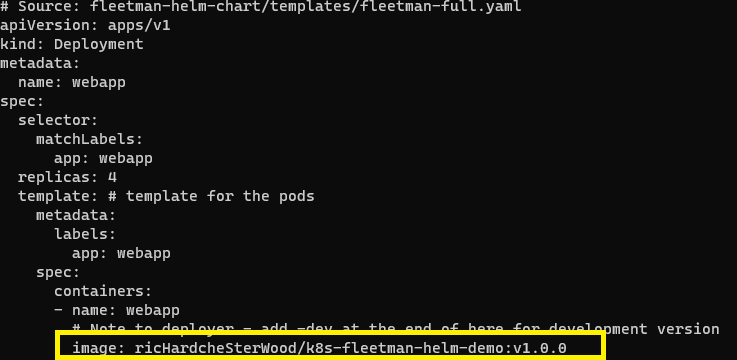
  noofreplica: 4

dockerRepo: "ricHardcheSterWood"



* Now Run the processing again will show wrong case Repository Name

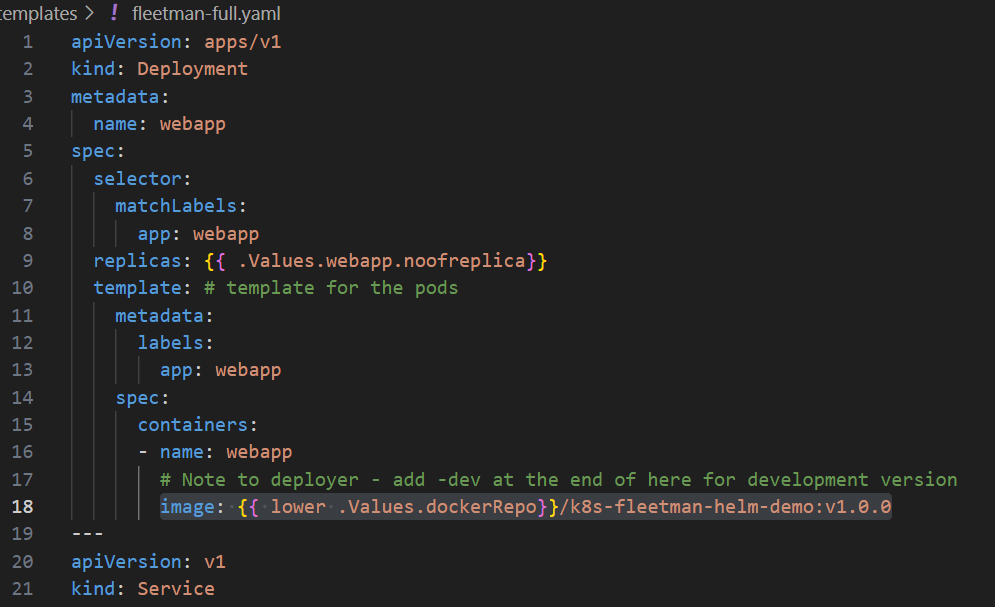
helm template .



* Now we want to defend against it , so we going to use **lower** function here.

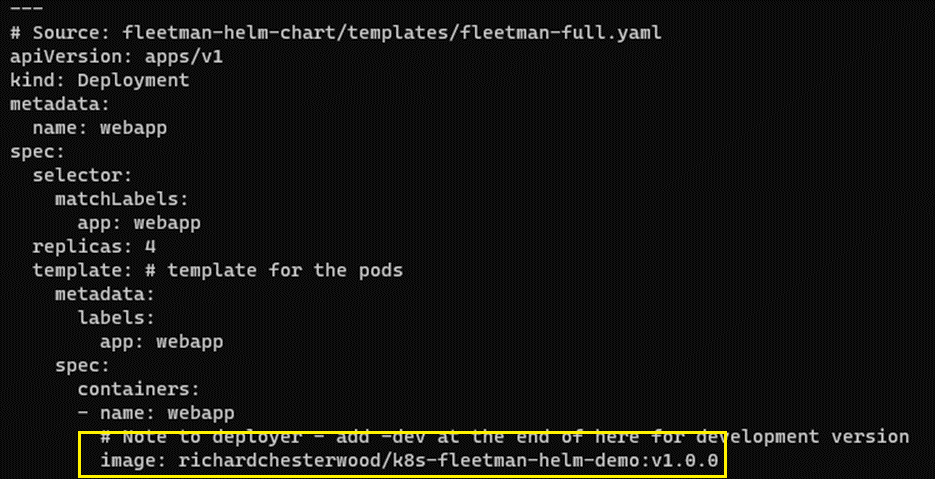
 In the **go language**, a function call looks like this. It's the name of the function, **followed by a space** and then a list of any of the parameters that you're passing in. And the parameters are **just space separated**. So there's no round brackets here.

image: {{ lower .Values.dockerRepo}}/k8s-fleetman-helm-demo:v1.0.0



* Now Run the processing again will show correct case Repository Name

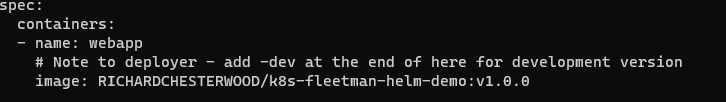
helm template .



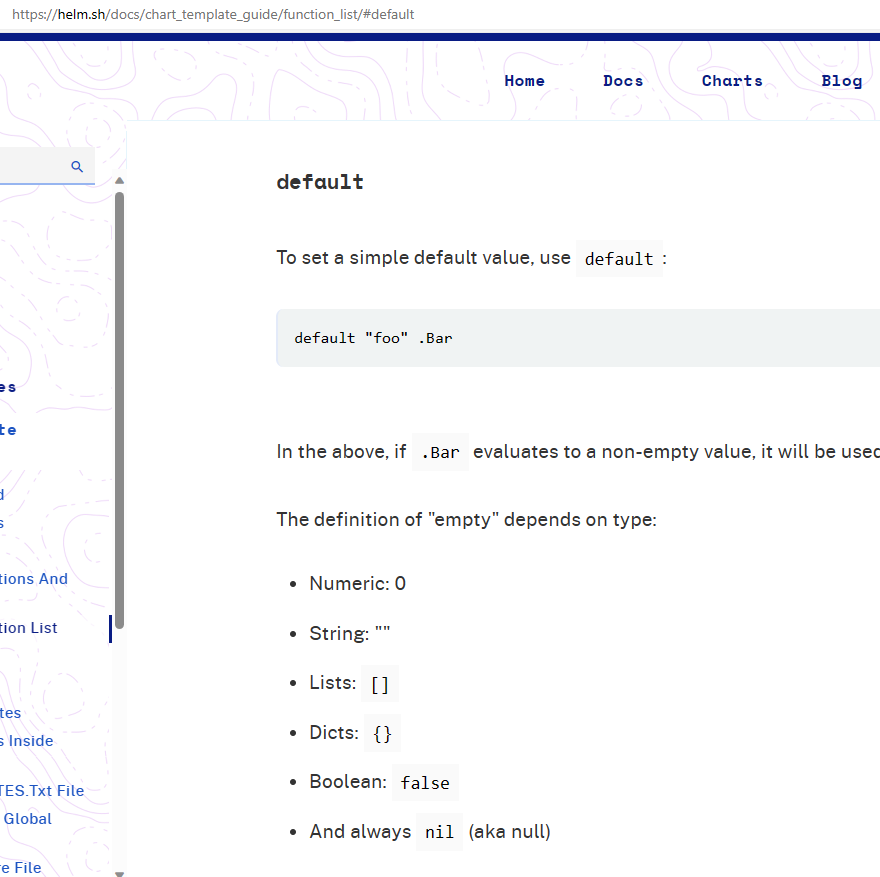
* Now try with upper function as well

image: {{ upper .Values.dockerRepo}}/k8s-fleetman-helm-demo:v1.0.0

* Running it again show repo name as upper case

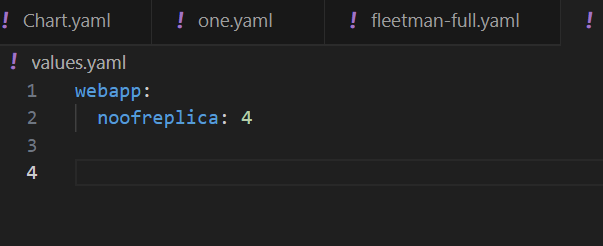


# Helm Functions (Default)



* What would happen if for some reason, there was no value for Docker repo name specified

in values dot yaml. Let's simulate that by removing the Docker repo name from our values dot yaml.

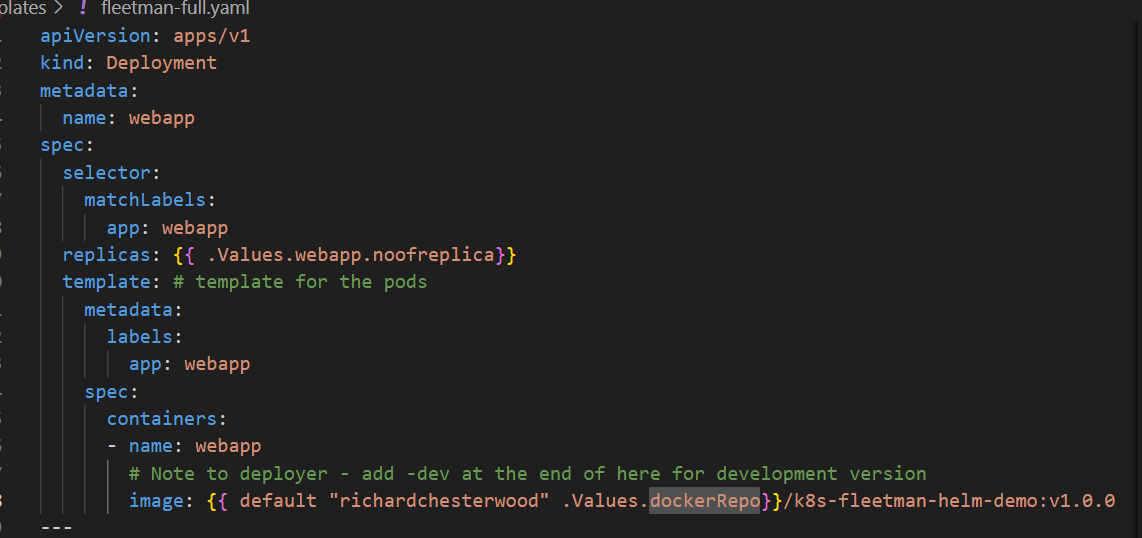


* Now running it will throw the error



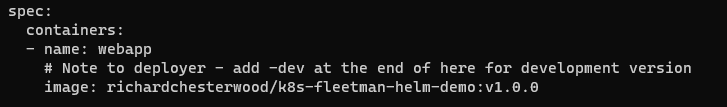
* Now what we could have done in the chart is we could defend against this, we could put into the code, The **default function** takes **two parameters**. The first parameter is what we would like to hard code in as a default value if none is provided by the end user and The second parameter is as before the pointer to the values in the value stock YAML file

image: {{ default "richardchesterwood" .Values.dockerRepo}}/k8s-fleetman-helm-demo:v1.0.0



* Now Run the processing again will show default Repository Name

helm template .

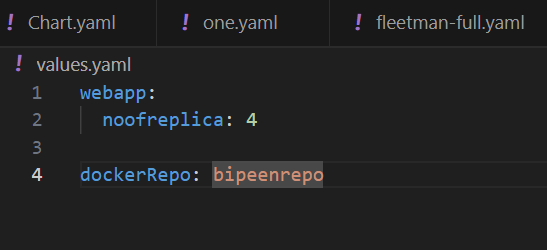


* Now lets restore the **docker repo** with **deferent name** in value.yaml

webapp:

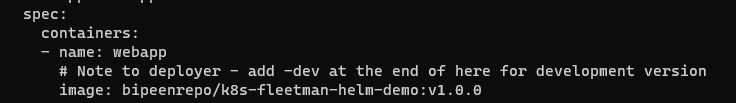
  noofreplica: 4

dockerRepo: bipeenrepo



* Now Run the processing again will show Value.yaml **Repository Name**

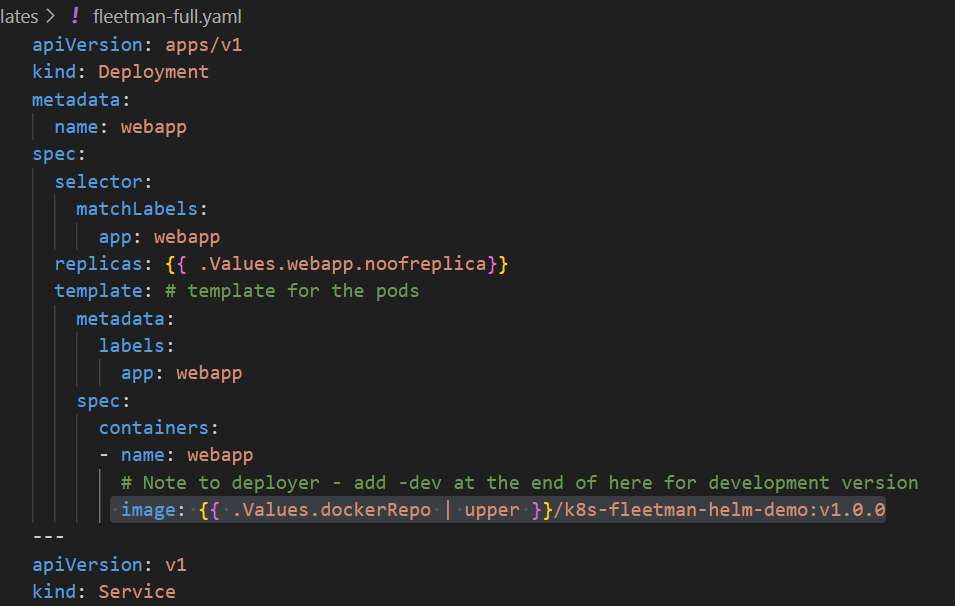
helm template .



# Helm Pipeline

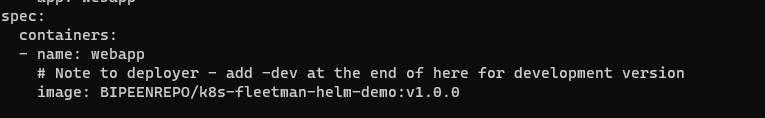
* Lets took the value of reponame and pass it to upper function using pipeline

 image: {{ .Values.dockerRepo | upper }}/k8s-fleetman-helm-demo:v1.0.0



* Now Run the processing again will show Value.yaml **Repository Name** in upper case

helm template .



# Helm Pipeline Chaining

image: {{ .Values.dockerRepo |default "richardchesterwood" | upper }}/k8s-fleetman-helm-demo:v1.0.0

---