# Kubernetes

# High level Architecture

# 

# Python Flask Application calling a Backend API which return the backend server host name which processed the request.

# Front End

# Implementation: - Download an image from web and copy to a shared folder for the front-end application to use it. Frontend application expect an image to be present under a specified directory to function.

# Inputs required to be provided for the front-end application are below

# Image: nevincleetus/frontend:v2

# Number of Replicas : 3

# Service Type : NodePort

# NodePort Port : Any Port between (30000 to 32K) For Exp: 30005

# :

# An environment variable to be configured in the POD definition to use from front end application.

# APPURL : For Exp: - <http://backend-service:5000>

# 

# Front end application shall invoke the back-end service using the above URL. The Cluster IP service URL shall be provided as through a ConfigMap to the Front end yaml file. The Front-end application reads the URL using environment variable (‘APPURL’).

# 

# Frontend application also expect an image name (“/app/static/cat.png”) before starting the application. The image needs to be downloaded and made available before starting the front-end application container.

# You may use the below URL to download the image. The front-end application POD shall use Init container to download and copy to shared volume.

# 

# URL "https://homepages.cae.wisc.edu/~ece533/images/cat.png"

# Hint: You may use below command to download and copy the image to required folder.

# image: busybox

# command:

# - wget

# - "-O"

# - "/<destination\_location>/cat.png"

# - "https://homepages.cae.wisc.edu/~ece533/images/cat.png"

# Configure a volume in your YAML file which can then be mounted to both Init container and the main front end container.

# Implementation details: -

# Create a deployment object to ensure 3 replicas from the docker image “nevincleetus/frontend:v2“ [Refer deployments in day1/yaml ]

# Create a NodePort service to select the matching pods with label as frontend as expose it as NodePort on any port within 3000-32K range

# Init Container to download the image (There will be 3 Init Containers running which will download the image. That is ok for now) and copy it to /app/static folder.

# [Refer pod in day1/yaml/pod ]

# Create a ConfigMap to configure the cluster IP URL for front end application to connect to back end. This need to be available as an environment variable ‘APPURL’ from the Front-end application. [ConfigMap in Day2 using environment variable]

# Backend Service

# Implementation: - Backend service shall return the hostname of the POD which processed the request. It also writes the time when the request was processed in a log under /app/logs directory. Currently a log file is created against each request. Use a hostPath volume to mount this path to a shared folder in the host path.

# Log file Cleaning: - We shall also create a cronjob which will be scheduled to run every 5 minutes to clean up the logs from the log directory which the Backend POD uses. Since we mount the logs file directory to a hostPath, the same shall be used from the cronjob to cleanup the log files under the folder.

# Inputs required to be provided for the back-end application are below

# Image: nevincleetus/backend:v2

# Number of Replicas : 3

# Service Type : ClusterIP

# NodePort Port : 5000

# Implementation details: -

# Create a deployment object to ensure 3 replicas from the docker image “nevincleetus/backend:v2“ [Refer deployments in day1/yaml ]

# Create a ClusterIP service to select the matching pods with label as backend as expose it as ClusterIP service. You may hardcode the ClusterIP if needed. But the same IP need to be provided in your ConfigMap which the Front end application uses. Use hostPath volume to mount the host path to logs directory [/app/logs/]

# Create a CronJob which will use the same volume as the backend POD and delete all log files from the directory.

# Expected Response from Front End application when accessed from the NodePort service from browser.

# <html>

# <body>

# <img src="/static/cat.png" />

# <h1>Frontend hostname... &#34;a35cc803e5df&#34; Backend hostname:... &#34;f3f5ac662d06&#34;<h1>

# </body>

# </html>