DDSM Application is hosted on GCP on a Kubernetes cluster.

The User Interface for DDSM. The User interface for displaying the classification results is developed in javascript, jquery, bootstrap and Flask.

DDSM is an application which interacts with models that are served on a Kubernetes Cluster on GCP using Kubeflow. DDSM uses Kubeflow, an ML toolkit for Kubernetes, to preprocess, hypertune, train, validate and serve the Deep Learning Models. DDSM leverages the essential elements of MLOps to train and serve the model.

The following are the steps involved with the entire development and deployment for DDSM.

Cluster Creation. Before we start on anything, there is a need for creating a Kubeflow cluster for developing the model. We create the Kubeflow cluster using CLI. The following are the steps which have to taken to create a Kubeflow cluster.

1. Download kfctl from the kfctl release page <https://github.com/kubeflow/kfctl/releases>
2. Select the resource file for the Kubeflow cluster. In our example we are using

CONFIG\_URI=<https://raw.githubusercontent.com/kubeflow/manifests/v1.0-branch/kfdef/kfctl_gcp_iap.v1.0.2.yaml>

1. Choose a Kubeflow cluster name i.e, KF\_NAME=ddsm-1208
2. Create a directory for downloading the installation artifacts

KF\_DIR=${BASE\_DIR}/${KF\_NAME}

mkdir ${KF\_DIR}

1. The cluster can be created by the following command

Kfctl apply -V -f ${CONFIG\_URI}

1. Once the cluster is created, the cluster can be launched by selecting the cluster end point
2. More information about Kubeflow is available <https://www.kubeflow.org/>

**Development environment**

The development environment is structured in a following manner. There are different stages of the model development which reflect in the different work areas

Kubeflow requires Docker images to be built for deploying them on a Kubeflow cluster. The directory structure for different stages of a model development pipeline follows the following guidelines

Folder

|--- Dockerfile

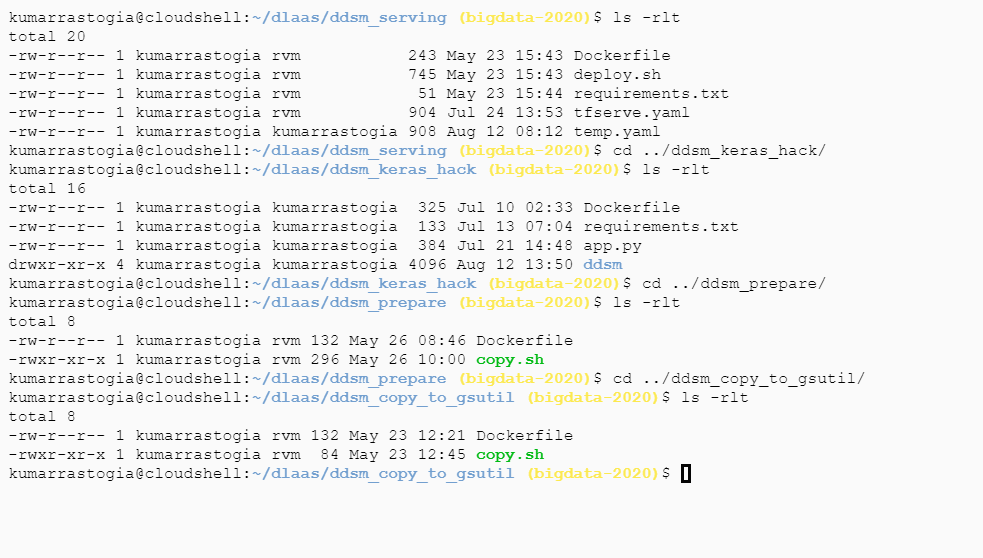
|--- requirements.txt

|--- <work dir>

The work directory contains the code for the stage of the pipeline it caters to. We have 5 stages in the MLOps pipeline .

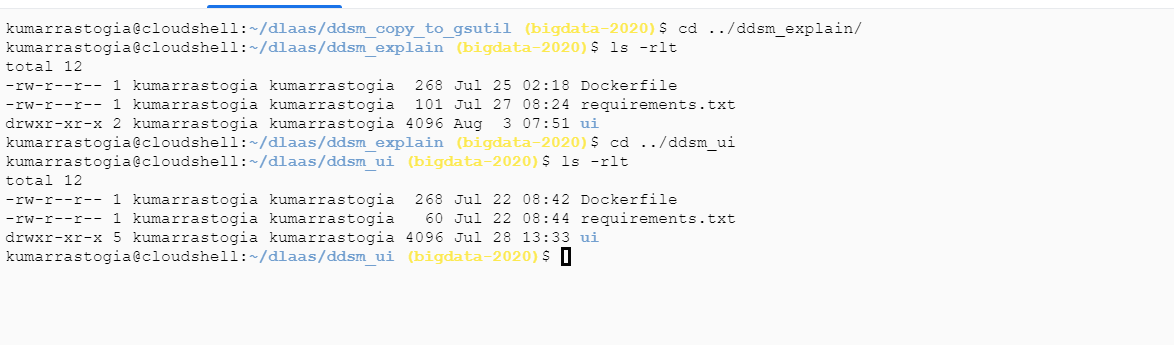
1. COPY TRAINING Data from GCP Storage :- This step copies the training data into a PVC attached to the Kubeflow/kubernetes cluster. All the relevant code for copying the training data and setting up the storage goes in <work dir>
2. Model Hyper parameter Tuning :- This step requires model hyper parameter tuning. Hyper parameter tuning reuses the model training code and hence the code is the same for model training.
3. Model Training – All the necessary code is present in the <model dir> directory. The docker image copies all the relevant python files into the docker image along with the necessary dependencies provided by requirements.txt file.
4. Model Serving – Model are served on TensorFlow serving framework. The code which goes with the same is in the <work dir>

Below is a directory structure



The ddsm UI and the ddsm Explain modules are also hosted on the Kubernetes cluster.

They also have a similar directory structure.



Docker images are built and pushed to docker registry (Google Container Registry)

The following command builds a docker image ddsm\_ui

docker build -t gcr.io/bigdata-2020/ddsm/ui ;

The following command pushes the docker image to GCR

docker push gcr.io/bigdata-2020/ddsm/explain/ui

Once the image is pushed, the GCR UI will look as below

