

# Analytics Edge Ecosystem Workloads - Google Summer of Code

Publication Date: 2022-09-11

## Contents

1	Introduction	2
2	Technical overview	3
3	Prerequisites	5
4	Install Rancher	5
5	Install Longhorn	5
6	Install Kubeflow Pipelines	7
7	Download the Data	8
8	Flask Interface	11
9	Machine Learning Model	13
10	Connect as an external client	16
11	Demonstration	17
12	Summary	17
13	Legal notice	19

# 1 Introduction

## 1.1 Motivation

`../../../../common/adoc/common_docinfo_vars.adoc`. Cancer is world's second-leading cause of death in the world. It results in development of abnormal cells that divide uncontrollably and have the ability to infiltrate and destroy normal body tissue. Eventually, survival rates are improving and all thanks to cancer screening, treatment and prevention.

Machine Learning is one of the aspect which can be integrated with the modern science and can create wonders. Under the Google Summer of Code, we at SUSE organization have created a Machine learning based Cancer Prediction Model for early screening.

This Document will demonstrate the detailed approach undertaken to accomplish this project. The project is developed under mentorship of Bryan Gartner, Ann Davis, Brian Fromme and the SUSE organization.

## 1.2 Scope

This guide introduces the basic concepts and steps to install, configure, and use the Cancer Prediction System in a SUSE Rancher Kubernetes environment. It uses Convolutional Neural Nets (CNNs) to automatically recognize the complex patterns in imaging data, providing quantitative as well as qualitative assessments of data within a short period of time.

## 1.3 Audience

This document is intended for Hospital Administrators, Radiologists, Doctors, Data Scientists, Data Engineers, and Data Analysts who are interested in using the Cancer Prediction System in a SUSE Rancher Kubernetes environment. It is also intended for SUSE Rancher users who want to learn how to install and configure the Cancer Prediction System.

## 2 Technical overview

As a part of GSoC, We have built a Machine Learning Based Cancer Prediction System. The fundamental goal of the system is the prediction of Cancer Susceptibility (also known as risk assessment), in this case, we are trying to predict the likelihood of developing a cancer prior to occurrence of the disease.

The Cancer Prediction System uses CT Scanned Images in the form of DICOM (Digital Imaging and Communications in Medicine) (<https://www.dicomstandard.org/>). The system is designed to be deployed as a Microservice based architecture and is divided into four interfaces:

- Lab Technician Dashboard
- Doctor Dashboard
- Rancher Dashboard
- ML Pipeline Dashboard

### 2.1 Components and tools

Key Cancer Prediction System components discussed in this guide are:

#### Flask Application

Flask is a micro web framework also written in Python. It is lightweight framework used to create web applications easily. We are using flask to serve the Backend and Machine Learning models as APIs.

#### Kubernetes Architecture

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. We have a kubernetes manifests designed to set up application over the cluster. Our application is compatible with various kubernetes distributions. We have tested our application on SUSE Rancher.






#### Keras

Keras is an open-source software library that provides a Python interface for designing artificial neural networks. It acts as an interface for TensorFlow library. We are using keras to design Convolutional Neural Nets (CNNs) (<https://www.tensorflow.org/tutorials/images/cnn>) for our application.

## Kubeflow Pipelines

It is used for machine learning pipelines to orchestrate workflows running on our Kubernetes clusters. Kubeflow allows us to focus on writing ML algorithms instead of managing their operations.

Some additional components and tools discussed in this guide include:

- Longhorn (<https://longhorn.io/>) : the cloud native distributed block storage system for Kubernetes.
- Rancher (<https://rancher.com/>) : the Kubernetes management platform.
- kubectl (<https://kubernetes.io/docs/reference/kubectl/>) : the command line tool for communicating with the Kubernetes cluster's control plane via the Kubernetes API.
- kustomize (<https://github.com/kubernetes-sigs/kustomize>) : a stand-alone tool for programmatically customizing Kubernetes objects.
- Docker (<https://www.docker.com/>) : the container runtime used to build and run the application.

## 2.2 Process overview

Getting started with Cancer Prediction System is fairly easy. In general, the process is as follows:

1. Clone the project repository: `git clone https://github.com/abhi-bhatra/ct\_image\_scanning.git`
2. Log in to your SUSE Rancher environment and select a managed cluster
3. Apply kubernetes manifests to setup the environment (*namespaces* and *storage*)
4. Apply kubernetes manifests to deploy the application
5. Access the application via the Rancher Dashboard

## 3 Prerequisites

This guide assumes that you have access to an existing Kubernetes cluster managed by [SUSE Rancher](https://rancher.com/docs/rancher/v2.6/en/overview/) (<https://rancher.com/docs/rancher/v2.6/en/overview/>) [↗](#). A good place to start getting more details is the [Deploying cluster Using Rancher Kubernetes Engine or K3S](https://medium.com/@abhinavsharma332/deploy-single-node-cluster-using-k3s-or-rke-6fc9e6a38b66) (<https://medium.com/@abhinavsharma332/deploy-single-node-cluster-using-k3s-or-rke-6fc9e6a38b66>) [↗](#) blog.



### Tip

Be sure docker is installed, as RKE (*Rancher Kubernetes Engine*) clusters need Docker as a prerequisite. Visit [Docker website](https://www.docker.com/get-started/) for more information (<https://www.docker.com/get-started/>) [↗](#).

## 4 Install Rancher

There are various ways to install the Rancher on the existing cluster. This section provides an overview of installing options of Rancher (<https://docs.ranchermanager.rancher.io/v2.5/pages-for-subheaders/installation-and-upgrade>) [↗](#).



### Tip

For this guide, we will be using the [High-availability Kubernetes Install with the Helm CLI](https://docs.ranchermanager.rancher.io/v2.5/pages-for-subheaders/install-upgrade-on-a-kubernetes-cluster) (<https://docs.ranchermanager.rancher.io/v2.5/pages-for-subheaders/install-upgrade-on-a-kubernetes-cluster>) [↗](#) to install Rancher on top of the cluster.

## 5 Install Longhorn

Before proceeding, review the concepts of persistent volumes, persistent volume claims (PVCs), and storage classes in the [Rancher documentation](https://rancher.com/docs/rancher/v2.6/en/cluster-admin/volumes-and-storage/how-storage-works/) (<https://rancher.com/docs/rancher/v2.6/en/cluster-admin/volumes-and-storage/how-storage-works/>) [↗](#).

Longhorn provides a storage solution that is easy to use, self-healing, and highly available. Longhorn is designed to be deployed on a Kubernetes cluster and provides persistent storage for stateful applications. There are 3 ways to installing Longhorn to Clusters:

1. Using the Apps and Marketplace in Rancher UI (<https://longhorn.io/docs/1.3.1/deploy/install/install-with-rancher/>) 
2. Using the kubectl manifests files (<https://longhorn.io/docs/1.3.1/deploy/install/install-with-kubectl/>) 

```
kubectl apply -f https://raw.githubusercontent.com/longhorn/longhorn/v1.2.4/deploy/longhorn.yaml
```

3. Using the Helm (<https://longhorn.io/docs/1.3.1/deploy/install/install-with-helm/>) :

```
helm repo add longhorn https://charts.longhorn.io
helm repo update
helm install longhorn/longhorn -name longhorn -namespace longhorn-system
```



## Tip

Cancer Prediction System strongly recommends using atleast two-worker nodes cluster.

## 5.1 Provision local storage

At this point, you can use Longhorn to format and manage the drives in your cluster nodes.

1. Provision the storage using the Flask Applications to share data and information on local cluster

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: data-pvc
  namespace: concerns
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

```
storageClassName: longhorn
```

## 2. Create a PersistentVolumeClaim (PVC) for the data volume

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ds-pvc
  namespace: concerns
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: longhorn
```

## 6 Install Kubeflow Pipelines

The Kubeflow project provides a straightforward way to deploy best-of-breed open-source systems for ML to diverse infrastructures. Anywhere you are running Kubernetes, you should be able to run Kubeflow. The [Kubeflow documentation \(https://www.kubeflow.org/docs/started/k8s/\)](https://www.kubeflow.org/docs/started/k8s/) provides a detailed guide to installing Kubeflow on Kubernetes.

We will look at how to deploy Kubeflow Pipelines standalone on our local clusters:

### 1. Set the PIPELINE\_VERSION and apply the Kubeflow Pipelines manifest:

```
export PIPELINE_VERSION=1.8.3

kubectl apply -k "github.com/kubeflow/pipelines/manifests/kustomize/cluster-scoped-resources?ref=$PIPELINE_VERSION"

kubectl wait --for condition=established --timeout=60s crd/applications.app.k8s.io

kubectl apply -k "github.com/kubeflow/pipelines/manifests/kustomize/env/platform-agnostic-pns?ref=$PIPELINE_VERSION"
```



## Note

It will take 15 to 20 mins to deploy the Kubeflow Pipelines on your cluster. You can check the status using `kubectl get all -n kubeflow`


2. Once all the services will start, you can see all pods status 1/1 Running. Your output will be somewhat similar to this:

NAME	READY	STATUS	
RESTARTS AGE			
pod/workflow-controller-5667759dd7-fbgrp 2d3h	1/1	Running	0
pod/ml-pipeline-scheduledworkflow-7f8bc78db9-qpx4f 2d3h	1/1	Running	0
pod/ml-pipeline-viewer-crd-8497d9695c-tqmdg 2d3h	1/1	Running	0
pod/ml-pipeline-ui-69bc756bd7-nmzm6 2d3h	1/1	Running	0
pod/metadata-envoy-deployment-6df8bdd989-lc77p 2d3h	1/1	Running	0
pod/minio-5b65df66c9-qt6lk 2d3h	1/1	Running	0
pod/ml-pipeline-persistenceagent-585c4b58d6-mcmtx 2d3h	1/1	Running	1
pod/ml-pipeline-7cc4f8fdf7-b2vjp 2d3h	1/1	Running	2
pod/cache-server-6cddb849-bnd6n 2d3h	1/1	Running	1

3. To access the Kubeflow Pipelines UI, you can use the following command:

```
kubectl port-forward -n kubeflow svc/ml-pipeline-ui 8080:80
```

## 7 Download the Data

The Cancer Prediction System uses the [CT Medical Images ~ cancer imaging archive with contrast and patient age \(https://www.kaggle.com/datasets/kmader/siim-medical-images\)](https://www.kaggle.com/datasets/kmader/siim-medical-images)  dataset from Kaggle.



The dataset is designed to allow for different methods to be tested for examining the trends in CT image data associated with using contrast and patient age. The basic idea is to identify image textures, statistical patterns and features correlating strongly with these traits and possibly build simple tools for automatically classifying these images when they have been misclassified (or finding outliers which could be suspicious cases, bad measurements, or poorly calibrated machines).

Cancer prediction system dataset directory system:

```
/dataset
-- archive/
  -- dicom_dir/
    .
    ID_0001_AGE_0069_CONTRAST_1_CT.dcm
    .
  -- tiff_images/
    .
    ID_0000_AGE_0060_CONTRAST_1_CT.tif
    .
  -- full_archive.npz
  -- overview.csv

-- dataset-classification
  -- Chest-CT/
  -- NonChest-CT/

-- dataset-prediction/
  -- train/
    -- cancer/
    -- non-cancer/
  -- test/
    -- cancer/
    -- non-cancer/
  -- validation/
    -- cancer/
    -- non-cancer/
```

1. **archive:** This folder comprises of raw dataset downloaded from Kaggle. We use python notebooks to process the data for further used in Machine Learning model.
2. **dataset-classification:** This is a separate dataset which separates all the DICOM Images as Chest and Non Chest. Currently, our model support Cancer classification on Chest DICOM Images. So, we need to filter out the Non Chest DICOM Images.
3. **dataset-prediction:** This is the final dataset used in Machine Learning model. All the raw images are processed into Train, Test and Validation sets. The labels are attached to the DICOM, so images can be classified as Cancer and Non-Cancer Images.

Apply the dataset deployment manifests to download the dataset from storage and process the data for further use in Machine Learning model.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: datasetvm
  namespace: cancerns
spec:
  replicas: 1
  selector:
    matchLabels:
      app: datasetvm
  template:
    metadata:
      labels:
        app: datasetvm
    spec:
      containers:
        - name: datasetvm
          image: "ubuntu:latest"
          imagePullPolicy: Always
          volumeMounts:
            - name: dataset
              mountPath: /dataset
          env:
            - name: DATASET
              value: "https://rancherdataset.blob.core.windows.net/ct-images/dataset.zip"
          command: ["/bin/sh", "-c"]
          args: ["apt-get update; apt-get install unzip wget -y; wget $DATASET -O /dataset/
dataset.zip; unzip /dataset/dataset.zip -d /dataset/dataset; ls -l /dataset/dataset"]
      volumes:
        - name: dataset
          persistentVolumeClaim:
```

## 8 Flask Interface

Complete Cancer Prediction System is built on top of Flask. It has two separate applications for the doctor and the radiologist. Directory Structure of the application is as follows:

```
/application
-- doctor_app/
  -- app.py
  -- Dockerfile
  -- classification-model.h5
  -- prediction-model.h5
  -- requirements.txt
  -- static/
    -- styles/
      -- css/
      -- js/
  -- template/
    -- base.html
    -- gallery.html
    -- predict.html
    -- retrain.html
    -- upload.html

-- lab_tech/
  -- app.py
  -- Dockerfile
  -- classification-model.h5
  -- adjust.py
  -- requirements.txt
  -- static/
    -- styles/
      -- css/
      -- js/
  -- template/
    -- base.html
    -- predict.html
    -- send.html
    -- upload.html
```

## 8.1 Deploy Lab Technician Interface

Lab Technician Interface is responsible for getting DICOM image as input. The person (*Radiologists, Lab Technician, Physicians*) could alter the information such as Contrast, Brightness and Angle of rotation of the DICOM image. They can also read all the information associated with the DICOM image (Modality: CT Scan).

Lab Technician Interface have a streamlined installation process. It can be easily installed through the Manifests.

1. Clone the repository

```
git clone https://github.com/abhi-bhatra/ct_image_scanning.git
```

2. Change the directory to k8s/lab-tech

3. Apply the kustomization file to deploy the Lab Technician Interface

```
kubectl apply -k .
```

4. To access the Lab Technician Interface, go to Rancher UI and click on the lab-tech service. Click on the Port button. Click on the port displayed. You will be redirected to the Lab Technician Interface.

## 8.2 Deploy Doctor Dashboard Interface

Doctor Dashboard is designed for the doctors to examine the report send by the Lab Technician. It receives the report of a patient and displays it to the user, predicting whether or not person is suffering from cancer. If doctor will not be satisfied with the response, they can send the image for the retraining with the correct label attached to it.

Just like Lab Technician Interface, doctor's dashboard Interface installation process is streamline and can be installed through manifests.

1. Clone the repository

```
git clone https://github.com/abhi-bhatra/ct_image_scanning.git
```

2. Change the directory to k8s/doctor-app/

3. Apply the kustomization file to deploy the Doctor Dashboard Interface

```
kubectl apply -k .
```

## 9 Machine Learning Model

In the project, Machine Learning is one of the major component used for prediction. We are using Convolutional Neural Network (CNN). It is a deep learning method and has achieved better results in detecting and segmenting specific objects in images in the last decade than conventional models such as regression, support vector machines or artificial neural networks.

### 9.1 Body-part Classification Model

Classification models are a subset of supervised machine learning. A classification model reads some input and generates an output that classifies the input into some category. In our case, model is taking CT-Scan and X-Ray images as input, and images are labelled. The model is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.

Steps for Classification model are as follows:

1. **Preprocessing:** In this step, we are converting the images into grayscale and resizing them to 128x128 pixels. We are also converting the labels into one-hot encoding.
2. **Splitting the dataset:** We are splitting the dataset into training, validation and testing sets. We are using 80% of the dataset for training, 10% for validation and 10% for testing.
3. **Building the model:** We are using Convolutional Neural Network (CNN) for building the model. We are using 3 convolutional layers and 2 dense layers. We are using Adam optimizer and categorical crossentropy loss function.
4. **Training the model:** We are training the model for 10 epochs. We are using 32 as the batch size.

Complete Reference to the Classification Model can be found at: [https://github.com/abhi-bhatra/ct\\_image\\_scanning/blob/UI\\_base/ML\\_MODEL/neural\\_net.ipynb](https://github.com/abhi-bhatra/ct_image_scanning/blob/UI_base/ML_MODEL/neural_net.ipynb) ↗

### 9.2 Cancer Prediction Model

In this model, we use machine learning in cancer diagnosis and detection. We are using Artificial neural networks (ANNs) for detecting and classifying tumors CRT images. We are using Convolutional Neural Network (CNN) for building the model. We are using 3 convolutional layers and 2 dense layers. We are using RMSprop optimizer and binary crossentropy loss function.

Complete Reference to the Cancer Prediction Model can be found at: [https://github.com/abhi-bhatra/ct\\_image\\_scanning/blob/UI\\_base/ML\\_MODEL/cancer\\_detection.ipynb](https://github.com/abhi-bhatra/ct_image_scanning/blob/UI_base/ML_MODEL/cancer_detection.ipynb) ↗

## 9.3 Retraining Model

Kubeflow Pipelines is a platform for building and deploying portable, scalable machine learning (ML) workflows based on Docker containers.

Retraining the Machine Learning Model is an automated process. We have used Kubeflow Pipelines to automate the process of retraining the model. It comprises of the following steps:

1. **Data Collection:** The data is collected from the dataset-prediction folder. The data is split into Train, Test and Validation sets. The labels are attached to the images. The data is then stored in the dataset folder.
2. **Data Preprocessing:** The data is preprocessed to remove the noise from the images. The images are converted into grayscale images. The images are then resized to 512x512. The data is then stored in the preprocessed folder.
3. **Trained Model:** The model is trained on the preprocessed data. The model is saved in the trained-model folder.
4. **Prediction:** The model is then converted into a prediction model. The prediction model is saved in the prediction-model folder.

Complete Reference to the Retraining Model can be found at: [https://github.com/abhi-bhatra/ct\\_image\\_scanning/blob/UI\\_base/ML\\_MODEL/kflow\\_model.py](https://github.com/abhi-bhatra/ct_image_scanning/blob/UI_base/ML_MODEL/kflow_model.py) ↗

## 9.4 Verify installation

After you have performed all of the above installation methods, validate that the Cancer Prediction System is installed and running.

1. List the pods running in the 'cancerns' namespace.

```
kubectll get all --namespace cancerns
```

This should produce output like the following:

NAME	READY	STATUS	RESTARTS	AGE
------	-------	--------	----------	-----

pod/datasetvm-5db64d7549-cflmf	1/1	Running	0	9s	
pod/doctor-app-d5b856997-64gnt	1/1	Running	0	10s	
pod/labtech-app-6c54f58874-jdmcw	1/1	Running	0	12s	

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/doctor-svc	NodePort	10.0.66.40	None	5002:30001/TCP	23s
service/labtech-svc	NodePort	10.0.145.57	None	5001:32009/TCP	24s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/datasetvm	1/1	1	0	13s
deployment.apps/doctor-app	1/1	1	0	23s
deployment.apps/labtech-app	1/1	1	0	23s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/datasetvm-5db64d7549	1	1	1	13s
replicaset.apps/doctor-app-d5b856997	1	1	1	10s
replicaset.apps/labtech-app-6c54f58874	1	1	1	21s

2. Verify that you see doctor-app- and labtech-app- pods in the 'Running' state and the service/doctor-svc and service/labtech-svc assigned to NodePort addresses.
3. Open the Kubeflow Console.

You can temporarily forward traffic from the Kubeflow Pipeline Console service to your local machine by issuing:

```
kubectrl port-forward -n kubeflow svc/ml-pipeline-ui 8080:80
```

This will produce output like the following:

```
Forwarding from 127.0.0.1:8080 -> 3000
Forwarding from [::1]:8080 -> 3000

Connect using: http://localhost:8080
```

4. To access the application in your Web browser, open the Rancher portal at <https://localhost:>



## Note

If you are using a different port on Rancher, put :PORT\_NUMBER with the port you are using.

## 10 Connect as an external client

By default, external applications cannot access the Cancer Prediction System.

With SUSE Rancher, you can set up either load balancers or ingress controllers to redirect service requests. Load balancers can only handle one IP address per service, while ingress works with one or more ingress controllers to dynamically route service requests.

It is recommended that you configure your cluster with an ingress.



### Note

Ingress and ingress controllers residing in RKE-launched clusters are powered by [NGINX](https://www.nginx.com/) (<https://www.nginx.com/>)<sup>7</sup>.

Below is a sample ingress resource for Cancer Prediction System:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-minio
  namespace: cancerns
  annotations:
    kubernetes.io/ingress.class: "nginx"
    ## Remove if using CA signed certificate
    nginx.ingress.kubernetes.io/proxy-ssl-verify: "off"
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/proxy-body-size: "0"
    nginx.ingress.kubernetes.io/server-snippet: |
      client_max_body_size 0;
    nginx.ingress.kubernetes.io/configuration-snippet: |
      chunked_transfer_encoding off;
spec:
  tls:
    - hosts:
        - cancerpred.example.com
      secretName: cancerpred-tls
  rules:
    - host: cancerpred.example.com
      http:
        paths:
          - path: /doctor
            pathType: Prefix
            backend:
```



```
    service:
      name: doctor-svc
      port:
        number: 443
  - path: /labtech
    pathType: Prefix
    backend:
      service:
        name: labtech-svc
        port:
          number: 443
```

## 11 Demonstration

Below is a demonstration of the Cancer Prediction System. The demonstration is divided into two parts:




1. Demonstration of the Cancer Prediction System as a whole.
2. Demonstration of the individual components of the Cancer Prediction System.

## 12 Summary

In this guide, you have explored how Artificial Intelligence is used for Cancer detection. This application has a streamline process of deploying into your SUSE Rancher Kubernetes landscape for consumption by your cloud native applications through a consistent API across all infrastructure platforms.

Below are a few resources to help you continue your exploration of Cancer Prediction System and SUSE Rancher:

- SUSE Technical Reference Documentation: Kubernetes (<https://documentation.suse.com/trd/kubernetes/>) ↗
- SUSE & Rancher Community (<https://www.suse.com/community/>) ↗
- Preparing for the next wave of transformation (<https://www.suse.com/c/preparing-for-the-next-wave-of-transformation/>) ↗
- Analytical Edge Ecosystem Worload for Healthcare Space ([https://github.com/abhi-bhatra/ct\\_image\\_scanning/tree/master](https://github.com/abhi-bhatra/ct_image_scanning/tree/master)) ↗

- Getting Started with Sample workload over Rancher (<https://medium.com/@abhinavsharma332/deploying-wordpress-over-rancher-cb9539b1d7da>) 
- Empowering Machine Learning Application on Rancher (<https://medium.com/@abhinavsharma332/empowering-machine-learning-applications-on-rancher-f4e368a9009>) 
- Orchestrate Machine Learning Model with Kubeflow (<https://medium.com/@abhinavsharma332/orchestrate-machine-learning-model-with-kubeflow-11945e7801b5>) 

## 13 Legal notice