Text Scanner

A Miniproject report submitted to the RGUKT-AP

in partial fulfillment of the degree of

Bachelor of Technology

in

Computer Science

By

Sindhuri Rudraraju

Rajiv Gandhi University of Knowledge Technologies

AP-IIIT, Rk Valley, Idupulapaya, Kadapa - 516 330

Andhra Pradesh, India

CERTIFICATE

This is to certify that the miniproject report entitled "**Text Scanner**" submitted by **Sindhuri Rudraraju**, bearing Id. No.R151303 , in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science is a bonafide work carried out by her under my supervision and guidance.
The miniproject report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Siva Rama Sastry Gumma
Project Supervisor
Dept of Computer Science,
RGUKT-AP.

Chandrasekhar N
Head of the Department
RGUKT-AP.

DECLARATION

I Sindhuri Rudraraju hereby declare that this Miniproject report entitled "**Text Scanner**" submitted by me under the guidance and supervision of **Siva Rama Sastry Gumma** is a bonafide work. We also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date: (Sindhuri Rudraraju)

Place:

To,

My Parents and My Teachers

Acknowledgments

I am extremely thankful and pay my gratitude to our faculty **Siva Rama Sastry Gumma** for his valuable guidance and support on completion of this project.

I am grateful to **N Chandrasekhar**, Head of the Department, RGUKT-AP for providing excellent facilities and a congenial atmosphere for progressing with our project.

I am grateful to **Rajiv Gandhi University of Knowledge Technologies** for providing all the necessary resources for the successful completion of my course work.

I also express my gratitude to my parents and family members who supported us morally.

At last but not least we would like to thank my friends who directly or indirectly helped me for this project.

With Sincere Regards,
Sindhuri Rudraraju

Abstract

This project's theme is to scan a text contained image and extract the text in it.Finding the text area is the first and important step, to do this we are using OpenCV's preprocessing steps .And algorithm used in the function to convert a preprocessed image as text is OCR(Optical character Recognition).OCR system converts scanned input document into editable text document. This report presents the detailed description about the characteristics of my Text Scanner system which uses OCR . The various stages of the Text Scanner system are:
upload a scanned image from the computer
segmentation process in which we extract the text zone from the image
recognition of the text and the last store both image and text using sqlite DB.
This report explains about the clean user interface provided with the help of which a user can add or modify scan results of Text scanner system very easily. And to make the application indipendent to platform and to make it secure Docker containerizing techtology is used so that dependency isues are no more. Finally to deploy the application Google's Google cloud platform(GCP) is used ,to maintein application more managable Google's kuberenetes Engine(GKE) is perfect solution.It provides an application Service discovery and load balancing,Storage orchestration Automated rollouts and rollbacks,Automatic bin packing,Self-healing,Secret and configuration management.Text Scanner also gives output for rotated text images.

List of Figures

Contents

Chapter 1

Introduction

In the running world, there is growing demand for the software systems to recognize charater in computer system when information is scanned through paper documents as we know that we have nubmber of newspapers and books which are in printed format related to different subjects. These days there is a huge demand in "storing the information available in these paper documents in to a computer storage disk and then later reusing this information by searching process".One simple way to sor information in these paper documents in to computer system is to fist scan the documents and then store them as IMAGES.But to reuse this informaton it is very difficult to read the individual contents and searching the contents from these documents line-by-line and word-by-word.The reason for this difficulty is the font characteristics of the characters in the paper documents are different to font of the characters in computer system.As a result,computer is unable to recognize the characters while reading them.This concept of storing the contents of paper document in computer storage place and then reading and searching the content is called DOCUMENT PROCESSING.Sometimes in this document processing we need to process the information that is related to languages other than the English in the world.For this document processing we can use Optical Character Recognition(OCR) a function module python .In my project also i am using python's openCV ,numpy,tesseract modules to process the image and recognize text in it.

1.1 OpenCV and OCR

1.1.1 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. In the Text Scanner project OpenCV was used for preprocessing the image which is very important to scan the image effectively and produce output correctly.

1.1.2 What is OCR

- Conversion of Images with Text into machine-encoded text
- Text in the image can be
  - Typed
  - Handwritten
  - Printed text
- Image can be
  - Scanned document
  - Photo of document
  - Scene photo
- It uses a python tool called Tesseract which actually can identify upto 116 languages

1.1.3 Applications of OCR

- Information extraction from documents
  - passport Recognition
  - Business Card information to contact list
  - Cheque/bank statements

- Number plate recognition
- Create text versions of printed documents and make them searchable
- Assistive technology(Can be used more for blind people)

1.2 Django

Django is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from doing scratch. Django uses MVC (Model-View-Controller) architechture which allows for code re-usability and high decoupling of the different layers The MVC architecture allows developers to change the visual part of an app and the business logic part separately, without their affecting one another. But actually, developers usually refer to Django's architecture as Model–View–Template (MVT). The three layers (Model, View, and Template) are responsible for different things and can be used independently. In Text Scanner i used Django to prepare a simple and clean web application from where user can give an image and get his output ,here user can also see all the scanned images so far and can delete them all. The beauty of it is that it follows the same philosophy as Python of "Batteries Included" (large library of useful modules) and so it comes with a set of tools off-the-shelf called Contrib Packages that make creating a fully featured and functional web apps a breeze. Examples include: authentication system, message framework, and admin interface amongst others. It is pre-configured to access a local SQLite Database which can easily be changed to something more powerful (like PostreSQL) with a simple-ish configuration change.As you develop and extend your application by adding or changing objects in your database (tables, columns, column types, foreign keys, ...) Django creates Migration scripts that can be applied to the destination Database, independently of the technology, by a simple command.Advantages of Django includes Rich ecosystem,Maturity,Admin panel by default,Good for SEO(Search engine optimization), Libraries,ORM(object relational mapper)

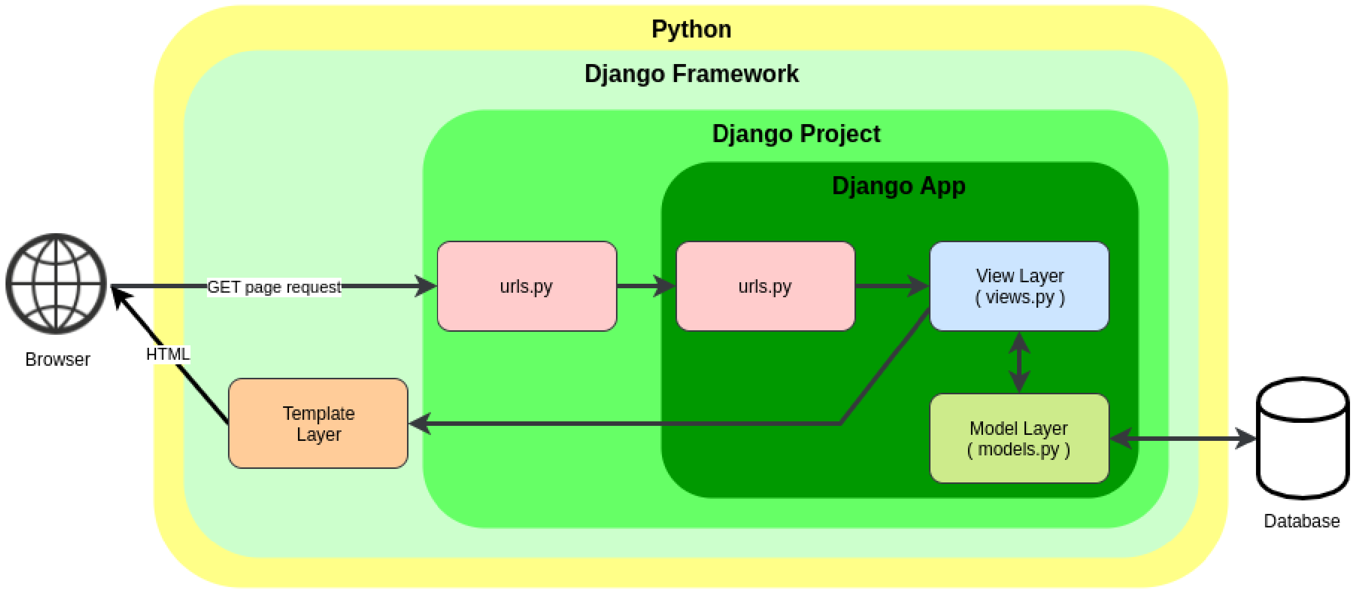


Figure 1.1: Simplified view of the Django components

1.3 Docker

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package.In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. Docker makes it easy for developers to develop and deploy apps inside neatly packaged virtual containerized environments. This means apps run the same no matter where they are and what machine they are running on. This gives a significant performance boost and reduces the size of the application. On the other hand, Docker communicates natively with the system kernel, bypassing the middleman on Linux machines, and even Windows 10, Windows Server 2016, and above. This means you can run any version of Linux in a container and it will run natively. Not only this, Docker uses less disk space too. So my Text scanner can be containerized with the all requirements and can be used by any other host without any prerequisites. And anyone can use the application just by pulling the Text scanner image.Before that the docker image should be created push to any docker registry.

This picture shows that containers are isolated but share OS and,where appropriate bins/libraries.

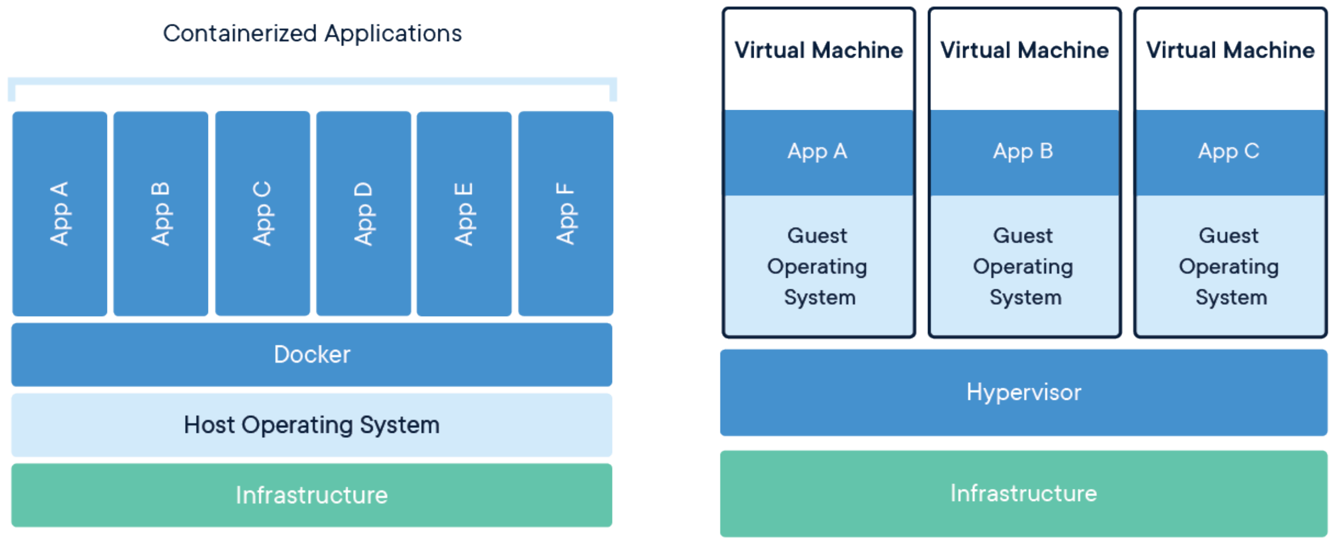


Figure 1.2: How docker is better from VMs

1.4 GCP and Kuberenetes Engine

Google Cloud Platform is a suite of public cloud computing services offered by Google. The platform includes a range of hosted services for compute, storage and application development that run on Google hardware. Google Cloud Platform services can be accessed by software developers, cloud administrators and other enterprise IT professionals over the public internet or through a dedicated network connection. GCP include

- Google Compute Engine: which is an infrastructure-as-a-service (IaaS) offering that provides users with virtual machine instances for workload hosting.
- Google App Engine: which is a platform-as-a-service (PaaS) offering that gives software developers access to Google's scalable hosting. Developers can also use a software developer kit (SDK) to develop software products that run on App Engine.
- Google Cloud Storage: which is a cloud storage platform designed to store large, unstructured data sets.Google also offers database storage options.

- Google Container Engine: which is a management and orchestration system for Docker containers that runs within Google's public cloud. Google Container Engine is based on the Google Kubernetes container orchestration engine.

For Text scanner, selection of vm-instances,memory needed are done by using Google Compute Engine. For storing the docker image created Google Container Engine is used ,Finally to deploy Textscanner as a microservice Google Kubernetes Engine is used (Using which one manage the applications based on user's traffic.

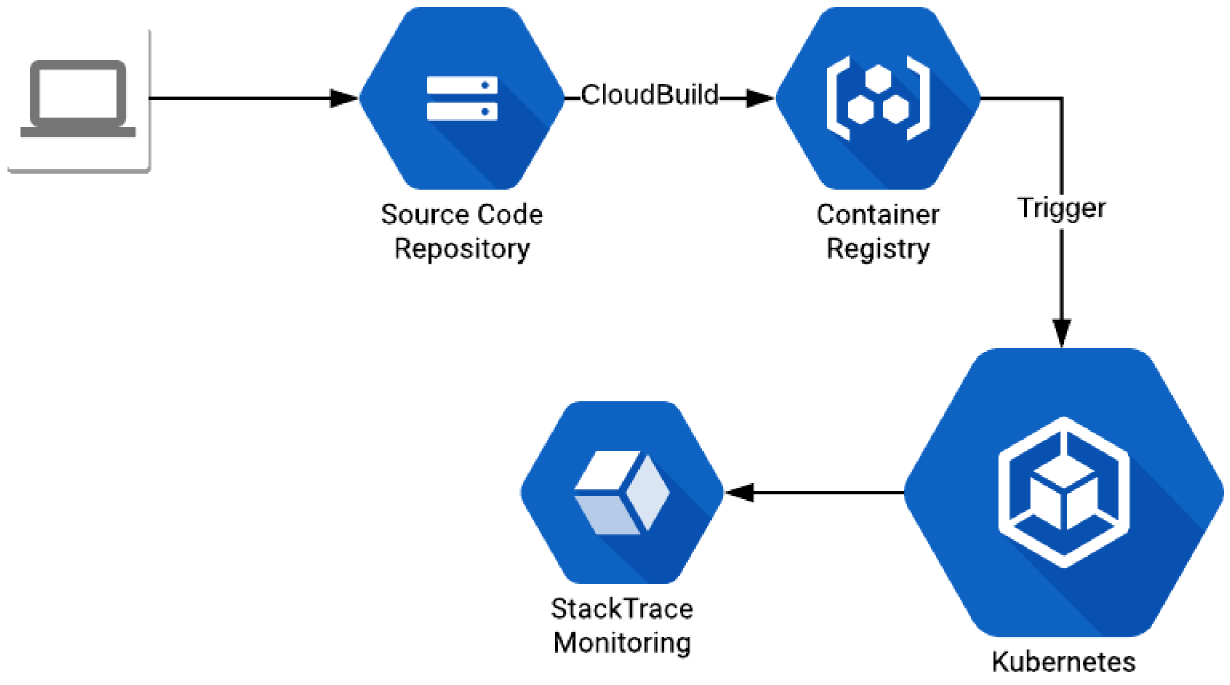


Figure 1.3: Kuberenetes work flow

Chapter 2

Code Overview

2.1 Code maintainance using Django including frontend and OpenCV libraries

Starting with Django using this, a simple and clean web application was preparing for textScanner.
Steps to setup application

2.1.1 Creating project and app in Django

- Start project: **django-admin startproject My_Text_Scanner** Look at what startproject created:

```
My_Text_Scanner/
        manage.py
        My_Text_Scanner/
              __init__.py
              settings.py
              urls.py
              asgi.py
              wsgi.py
```

In this directory structure

manage.py: A command-line utility that lets you interact with this Django project in various ways.

My_Text_Scanner/__init__.py:An empty file that tells Python that this directory should be considered a Python package.

My_Text_Scanner/settings.py:Settings/configuration for this Django project.

My_Text_Scanner/urls.py:The URL declarations for this Django project; a "table of contents" of our Django-powered site.

My_Text_Scanner/asgi.py:An entry-point for ASGI-compatible web servers to serve your project. My_Text_Scanner/wsgi.py:An entry-point for WSGI-compatible web servers to serve your project.

- Create app: **python manage.py startapp scanner**

  That'll create a directory scanner, which is laid out like this:

```
scanner/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

  This directory structure will house the scanner application.Anyway we are not going to modify all files but our required ones.

Now our application setup is ready ,let us modify files for our Text Scanner.

2.1.2 Setting media storage and creating urls

The very first step is to add MEDIA_ROOT and MEDIA_URL to settings.py file as we are storing and retriving images.

MEDIA_ROOT is for server path to store files in the computer. **MEDIA_URL** is the reference URL for browser to access the files over Http.

```
//settings.py


MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

MEDIA_URL = '/media/'
```

Also in the urls.py configuration should be added as per the image urls we have edited in settings.py

```
//urls.py
   if settings.DEBUG:
       urlpatterns += static(settings.MEDIA_URL,
                             document_root=settings.MEDIA_ROOT)
```

2.1.3 Database model and form creation

In Text scanner we are creating a database table with two fields Image and Text,Text is scanned text with Text Scanner and Image is which given by the user.So we are creating a model Scanner* with input_Image and image_Text as fields.

```
//models.py
from django.db import models
class Scanner(models.Model):
    input_Image = models.ImageField(upload_to='images/')
    image_Text = models.CharField(max_length=1000)
```

Here upload_to will specify, to which directory the images should reside, by default django creates the directory under media directory which will be automatically created when we upload an image. No need of explicit creation of media directory.To deal with model we created ,we are creating form.py* fle under scanner* to make content easier to understand.

```
//form.py
from django import forms
from .models import *

class ImageForm(forms.ModelForm):

    class Meta:
        model = Scanner
        fields = ['input_Image']
```

Django will implicitly handle the form verification's with out declaring explicitly in the script, and it will create the analogous form fields in the page according to model fields we specified in the models.py file. This is the advantage of model form.

2.1.4 View creation and home page includes Opencv packages

Now we create a **templates** directory under scanner* in that we have to create a html file for uploading the images.

```
//upload.html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Give image to scan</title>
</head>
<style>
```

```
    form {
  text-align: center;
    }
    table{
    width:100%;}
    h1 {text-align: center;}
    h4 {text-align: center;}
    .button2 {background-color: #ec584d;}


    .button1 {background-color: #ca56e7;}


</style>
<body style="background-color:rgb(202, 126, 74);">
    <table style="width:100%">
        <tr>
          <h1 style="color:rgb(193, 193, 224);">Welcome to my text scanner</h1>
        </tr>
    </table>
    <h4 style="color:blue;">Scan your image</h4>
    <tr>
    <form method = "post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button class="button1" type="submit" value="Scan" name="_scan">Upload
            your image</button>
    </form>
    <br><br><br>
    <h4 style="color:blue;">Get all scans</h4>
    <form method = "post" >
```

```
        {% csrf_token %}
        <button class="button1" type="submit" value="Get_scans"
           name="_getscans">Get scans</button>
    </form>
    <br><br><br>
    <h4 style="color:blue;">Delete all scans</h4>
    <form method = "post" >
        {% csrf_token %}
        <button class="button2" type="submit" value="Delete_scans"
           name="_delscans">Delete all scans</button>
    </form>
</body>
</html>
```

When making a POST request, we have to encode the data that forms the body of the request in some way. So, we have to specify the encoding format in the form tag. multipart/form-data is significantly more complicated but it allows entire files to be included in the data.The csrf_token is for protection against Cross Site Request Forgeries.form.as_p simply wraps all the elements in HTML paragraph tags. The advantage is not having to write a loop in the template to explicitly add HTML to surround each title and field.Using this html we have three different options for user to Input an image for scanning,to see all the scanned images and finally to delete all scanned images.

To take request from user and to give back some UI views.py* under scanner* will be used

```
//views.py
from django.http import HttpResponse
from django.shortcuts import render, redirect
from .form import *
from .models import Scanner
import cv2 as cv
import numpy as np
import pytesseract as pyt
from matplotlib import image
import os
import cv2 as cv
import numpy as np
import pytesseract as pyt
# we can create our own views here
def take_image(request):
    if request.method == 'POST':
        if '_getscans' in request.POST:
            return redirect('All_images')
        if '_delscans' in request.POST:
            return redirect('deleted')
        form = ImageForm(request.POST, request.FILES)


        if form.is_valid():
            form.save()
            obj=Scanner.objects.last()
            path = os.getcwd()
            parent = os.path.join(path, os.pardir)
            parent=parent.replace("..","")
            img = parent + obj.input_Image.url
          text = scan_Image(img)
            obj.image_Text=text
            obj.save()
            return redirect('display')
    else:
        form = ImageForm()
    return render(request, 'upload.html', {'form' : form})
```

```
def display_images(request):

    if request.method == 'GET':

        # getting all the objects of scanner
        Images = Scanner.objects.all()
        return render(request, 'display.html', {'all_images' : Images})
def display_image(request):
    if request.method == 'GET':

        # getting last object of scanner
        Image = Scanner.objects.last()
        return render(request, 'display_single.html', {'image' : Image})
def delete_images(request):
    if request.method == 'GET':

        # getting all the objects of hotel.
        Image = Scanner.objects.all().delete()
        return render(request, 'delete.html')


def scan_Image(img):
    img = cv.imread(img)
    #to white text on black background
    gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
    gray = cv.bitwise_not(gray)
    ret,thresh = cv.threshold(gray,0,255,cv.THRESH_BINARY)
    #Find the rect that bounds the text
    coords = np.column_stack(np.where(thresh>0))

    rect = cv.minAreaRect(coords)

    angle = cv.minAreaRect(coords)[-1]

    #Determine angle of inclination

    if angle < -45:

        angle = -(90 + angle)

    else:

        angle = -angle

    #Get the affine transform

    (h,w) = img.shape[:2]

    center = (w//2,h//2)

    M = cv.getRotationMatrix2D(center,angle,1.0)

   rotated =

       cv.warpAffine(img,M,(w,h),flags=cv.INTER_CUBIC,borderMode=cv.BORDER_REPLICATE)

    text = pyt.image_to_string(rotated)

    return text
```

When user gives an image and click ”upload your image” ,we are first saving that into DB and sending it to scan_image function where image will be preprocessed and outputs the text in it using OCR and the same text is saved to DB field then a UI with given image and extracted text will be displayed to the user.
when user click ”Get scans” ,we are getting all the objects of scanner* model and diplaying each image with corresponding text in a nice html page.
when user click ”Delete all scans”, we are deleting all objects of scanner* model in the DB and redirecting user to a successfull deleted html page.
Now look at the responsive html codes

```
\\display_single.html


<!DOCTYPE html>

<html lang="en">
```

```
<style>
    img {
        display: block;
        margin-left: auto;
        margin-right: auto;
        }

    div.justified {
        display: flex;
        justify-content: center;
    }
    h2 {text-align: center;}

</style>
<head>
    <meta charset="UTF-8">
    <title>your given image</title>
</head>
<body style="background-color:rgb(16, 192, 169);">
    <h2 style="color:rgb(34, 41, 5);">Image you have given</h2>
    <img src="{{ image.input_Image.url }}" style="width: 50%" />
    <h2 style="color:rgb(34,41,5);">Text scanned</h2>
    <div class="justified">
        <textarea rows="9" cols="100">
            {{ image.image_Text }}
            </textarea>
    </div>
</body>
</html>
```

```
//display.html


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Uploaded Images</title>

      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet"
          href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
      <script
          src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
      </script>
      <script
          src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js">
      </script>
</head>
<body>

    {% for image in all_images %}
            <div class="col-md-4">
                    <img src="{{ image.input_Image.url }}"
                        class="img-responsive" style="width: 100%; float: left;
                        margin-right: 10px;" />
                      <textarea rows="9" cols="50">
                          {{ image.image_Text }}
                          </textarea>
             </div>
    {% endfor %}


</body>

</html>
```

```
//delete.html


<!DOCTYPE html>

<html lang="en">

    <style>

        h2 {text-align: center;}

    </style>

<body style="background-color:rgb(113, 170, 192);">

    <h2 style="color:rgb(177, 21, 68);">Deleted_All_Images_successfully</h2>

</body>

</html>
```

To establish a connection between views.py* to redirecting pages we are adding url-patterns to urls.py* from which Django server can know what to do with the particular url.The empty path here shows the root/Home page.

```
\\urls.py

urlpatterns = [

    path('', take_image, name = 'image_upload'),

    path('All_images', display_images, name = 'All_images'),

    path('display', display_image, name = 'display'),

    path('deleted', delete_images, name = 'deleted'),

]
```

2.1.5 Migrations and running server

Now make migrations and the server

- **python manage.py makemigrations:** This will create and manage all our models creates DB (we can seleted preferable DB sqlite3,mysql,postgres,oracle...etc default is sqlite3)
- **python manage.py runserver <port_num>** This will run our application at the given port number ,if not specified 8000 is default.

Try accessing **localhost:<port_num>**

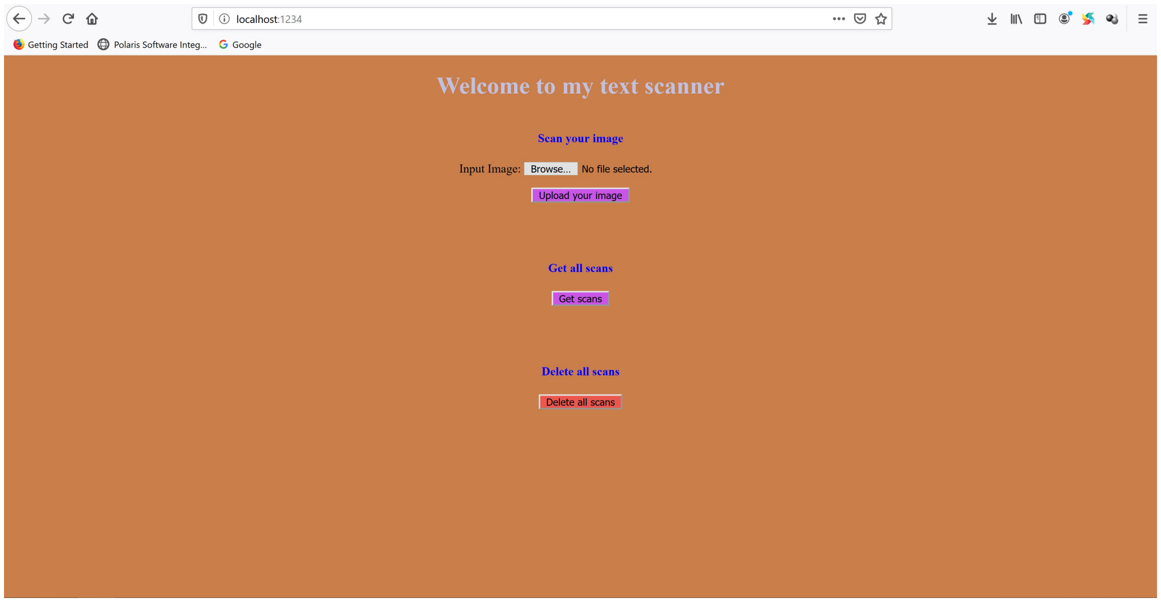


Figure 2.1: We get our output

Chapter 3

Deploying the application

Application Deployments define the package of software components that make up an application in a particular environment, e.g. development or production. Instances of these are deployed onto physical Technology Nodes to capture where that software is executing. Although this is technically an activity in the Technology Layer, it is the logical conclusion of the Application Layer working from the Conceptual, through Logical and the Physical Views.

3.1 Dockerizing Text Scanner

Dockerizing or Containerizing means wrapping all the requirements and source code as a bundle including specific versions of dependencies used for Text Scanner and making it available to any other host by making it available in any public or private registeries.

3.1.1 Build Text Scanner image

To build a docker image for our application we take a base image which is suitable ,from docker hub we can find many images which are open source like ubuntu,debian,alpine,python,jdk,tomcat,g ...many other .This base image will make a first layer for the image ,then we add all dependencies we need for the application ,for Text scanner mainly we need python environment where Django,OpenCV,numpy,pytesseract are installed .Each line we specified will build separate layer and our final image will be produced.Then we add the our applications source code to the image ,finally start our server inside image.

```
\\Dockerfile
FROM python:3.8.3
ENV PYTHONUNBUFFERED 1
RUN pip3 install Django==3.0.6
RUN pip3 install numpy matplotlib
RUN pip3 install opencv-python
RUN pip3 install pillow
RUN pip3 install pytesseract
RUN apt-get update \
    && apt-get install tesseract-ocr -y \
    python3 \
    python3-pip
COPY . /code/
WORKDIR /code
EXPOSE 8000
CMD /code/script.sh
```

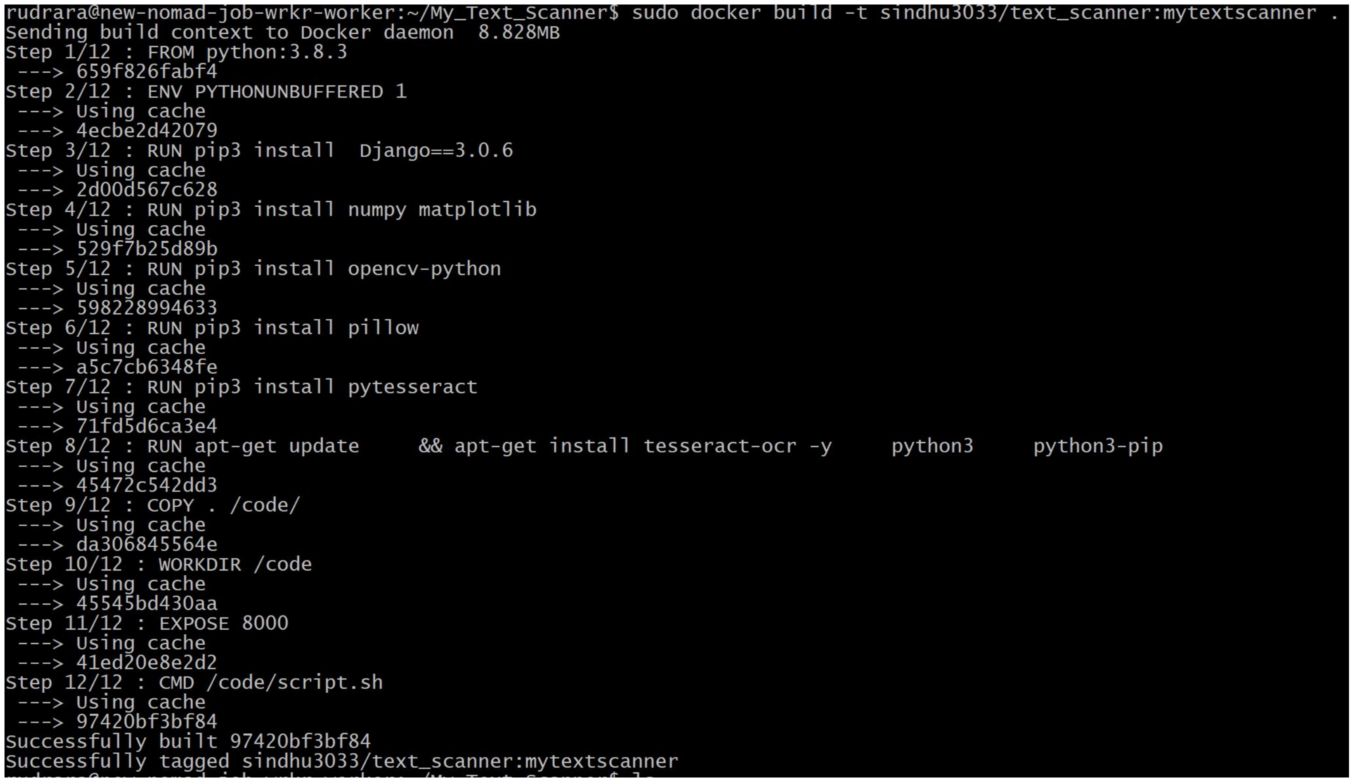
We have seleted a opensource base image python:3.8.3 which includes debian operating system which is a flavour of linux.Then using COPY* instruction we are copything all the source code to the image,setting the WORDIR(workdirectory) ,exposing 8000 port to be available and we are writing a script for the steps to run our server in scrip.sh*.

```
\\script.sh
#!/usr/bin/env bash
python manage.py migrate
echo "server running at port 8000"
python manage.py runserver 0.0.0.0:8000
```

Just two commands are required to run the server.Most importantly our docker file should be in the name Dockerfile * ,then only docker will recognize the image layers.Its time to build the image.

- **docker build -t sindhu3033/text_scanner:mytextscanner .**

  -t represents creating our own image ,myscanner* is the name of our image and ”.” tells docker to perform actions in the current directory.Image will be created.

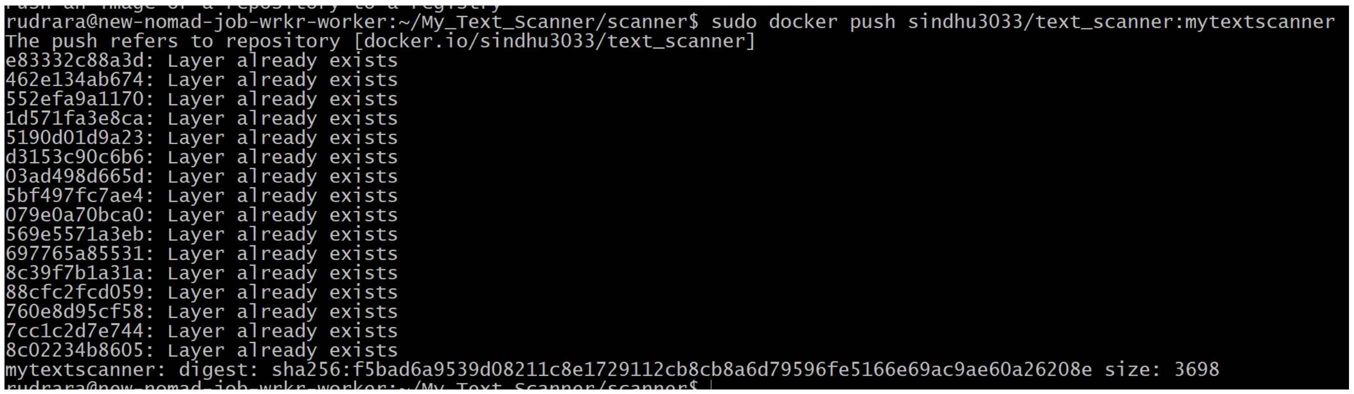


```
rudrara@new-nomad-job-wrkr-worker:~/My_Text_Scanner$ sudo docker build -t sindhu3033/text_scanner:mytextscanner .
Sending build context to Docker daemon  8.828MB
Step 1/12 : FROM python:3.8.3
 ---> 659f826fabf4
Step 2/12 : ENV PYTHONUNBUFFERED 1
 ---> Using cache
 ---> 4ecbe2d42079
Step 3/12 : RUN pip3 install  Django==3.0.6
 ---> Using cache
 ---> 2d00d567c628
Step 4/12 : RUN pip3 install numpy matplotlib
 ---> Using cache
 ---> 529f7b25d89b
Step 5/12 : RUN pip3 install opencv-python
 ---> Using cache
 ---> 598228994633
Step 6/12 : RUN pip3 install pillow
 ---> Using cache
 ---> a5c7cb6348fe
Step 7/12 : RUN pip3 install pytesseract
 ---> Using cache
 ---> 71fd5d6ca3e4
Step 8/12 : RUN apt-get update     && apt-get install tesseract-ocr -y     python3     python3-pip
 ---> Using cache
 ---> 45472c542dd3
Step 9/12 : COPY . /code/
 ---> Using cache
 ---> da306845564e
Step 10/12 : WORKDIR /code
 ---> Using cache
 ---> 45545bd430aa
Step 11/12 : EXPOSE 8000
 ---> Using cache
 ---> 41ed20e8e2d2
Step 12/12 : CMD /code/script.sh
 ---> Using cache
 ---> 97420bf3bf84
Successfully built 97420bf3bf84
Successfully tagged sindhu3033/text_scanner:mytextscanner
```

Figure 3.1: Building docker image

- **docker push sindhu3033/text_scanner:mytextscanner**

  This will push our image to docker public registry ”tex_scanner” with the user name ”sindhu3033”, now anyone can pull the image.



```
rudrara@new-nomad-job-wrkr-worker:~/My_Text_Scanner/scanner$ sudo docker push sindhu3033/text_scanner:mytextscanner
The push refers to repository [docker.io/sindhu3033/text_scanner]
e83332c88a3d: Layer already exists
462e134ab674: Layer already exists
552efa9a1170: Layer already exists
1d571fa3e8ca: Layer already exists
5190d01d9a23: Layer already exists
d3153c90c6b6: Layer already exists
03ad498d665d: Layer already exists
5bf497fc7ae4: Layer already exists
079e0a70bca0: Layer already exists
569e5571a3eb: Layer already exists
697765a85531: Layer already exists
8c39f7b1a31a: Layer already exists
88cfc2fcd059: Layer already exists
760e8d95cf58: Layer already exists
7cc1c2d7e744: Layer already exists
8c02234b8605: Layer already exists
mytextscanner: digest: sha256:f5bad6a9539d08211c8e1729112cb8cb8a6d79596fe5166e69ac9ae60a26208e size: 3698
```

Figure 3.2: Pushing docker image

- **docker pull sindhu3033/text_scanner:mytextscanner**
  This will pulls the image to the host machine(User should have docker installed in his local.)

- **docker run -it -p 1234:8000 sindhu3033/text_scanner:mytextscanner**
  This will port-forward the 8000 port from the dockercontainer created to our local port 1234.Now try accessing localhost:1234
  Our application has been dockerized successfully.

3.2 Deploying text scanner to kuderenetes engine

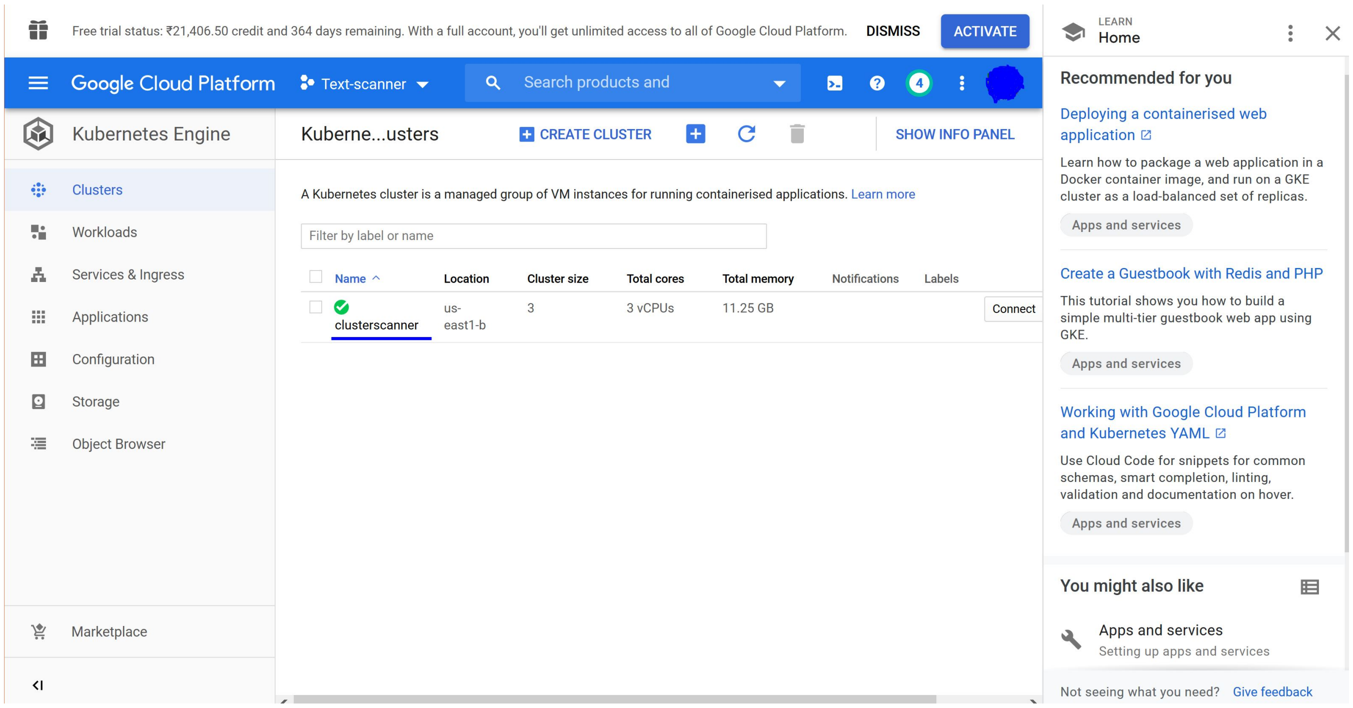
To make our application available over network kuberenetes is very scalable and robust solution.First of all create an account in Google Cloud Platform one can start with one month free tier .Created a project called Text-scanner,enabled Kuberenetes engine API under APIs & Services section .
Kuberenetes allows to create resources called deployments,services,jobs,cron jobs and delete resources also to manage them on the fly(editing deployments,jobs at any time).There are many useful cli commands to run our applications in our own way.
Requirements:

- **Gcloud cli**

- **Kubectlt binary**

To connect Text-scanner project to installed cli authenticate using **gcloud auth login** command ,select the nearest region ,zones and the project to work on.
Let us have look at simple steps to run application in kuberenetes.

- **gcloud container cluster create clusterscanner**
  This will create cluster named clusterscanner* which we can use to create wanted nodes for the application.This action will reflect in the console also.



Free trial status: ₹21,406.50 credit and 364 days remaining. With a full account, you'll get unlimited access to all of Google Cloud Platform.

A Kubernetes cluster is a managed group of VM instances for running containerised applications. Learn more

Filter by label or name

Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels	
✓ clusterscanner	us-east1-b	3	3 vCPUs	11.25 GB			Connect

Recommended for you

Deploying a containerised web application

Learn how to package a web application in a Docker container image, and run on a GKE cluster as a load-balanced set of replicas.

Apps and services

Create a Guestbook with Redis and PHP

This tutorial shows you how to build a simple multi-tier guestbook web app using GKE.

Apps and services

Working with Google Cloud Platform and Kubernetes YAML

Use Cloud Code for snippets for common schemas, smart completion, linting, validation and documentation on hover.

Apps and services

You might also like

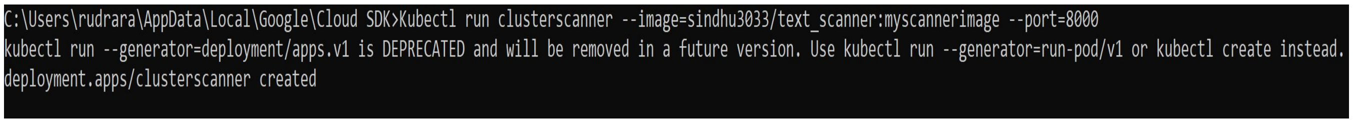
Apps and services
Setting up apps and services

Not seeing what you need? Give feedback

Figure 3.3: Cluster has been created at Google Kuberenetes Engine(GKE)

- **Kubectlt run clusterscanner –image=sindhu3033/text_scanner:myscannerimage –port=8000**

  This will run a deployment with our continerized docker image and run on port on 8000

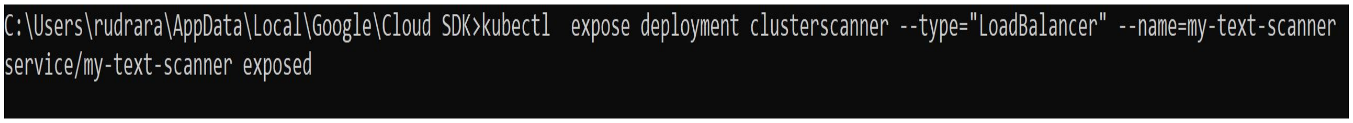


```
C:\Users\rudrara\AppData\Local\Google\Cloud SDK>Kubectl run clusterscanner --image=sindhu3033/text_scanner:myscannerimage --port=8000
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/clusterscanner created
```

Figure 3.4: Start running the created cluster

- **kubectl expose deployment clusterscanner –type="LoadBalancer" –name=my-text-scanner**

  Now textscanner is exposed to the internet world ,any user in the internet can access textscanner ,here type* options scecifies kuberenetes that if user traffic increased create another node for the app and manage.

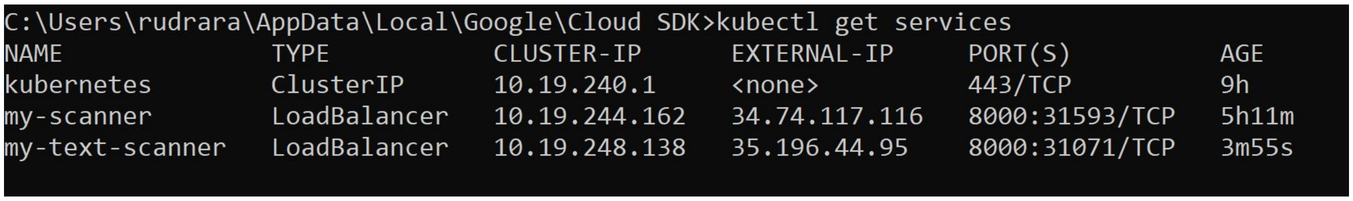


```
C:\Users\rudrara\AppData\Local\Google\Cloud SDK>kubectl  expose deployment clusterscanner --type="LoadBalancer" --name=my-text-scanner
service/my-text-scanner exposed
```

Figure 3.5: Exposing the kuberenetes deployment to the internet

- **kubectl get services**

  With this command the host address in which our created deployment running is known so that one can access over network.



```
C:\Users\rudrara\AppData\Local\Google\Cloud SDK>kubectl get services
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)          AGE
kubernetes        ClusterIP      10.19.240.1     <none>          443/TCP          9h
my-scanner        LoadBalancer   10.19.244.162   34.74.117.116   8000:31593/TCP   5h11m
my-text-scanner   LoadBalancer   10.19.248.138   35.196.44.95    8000:31071/TCP   3m55s
```

Figure 3.6: Services with their external and running IP addresses

Now take the external IP of the exposed service my-text-scanner* which is 35.196.44.95 along with the port running,and try accessing from any browser.

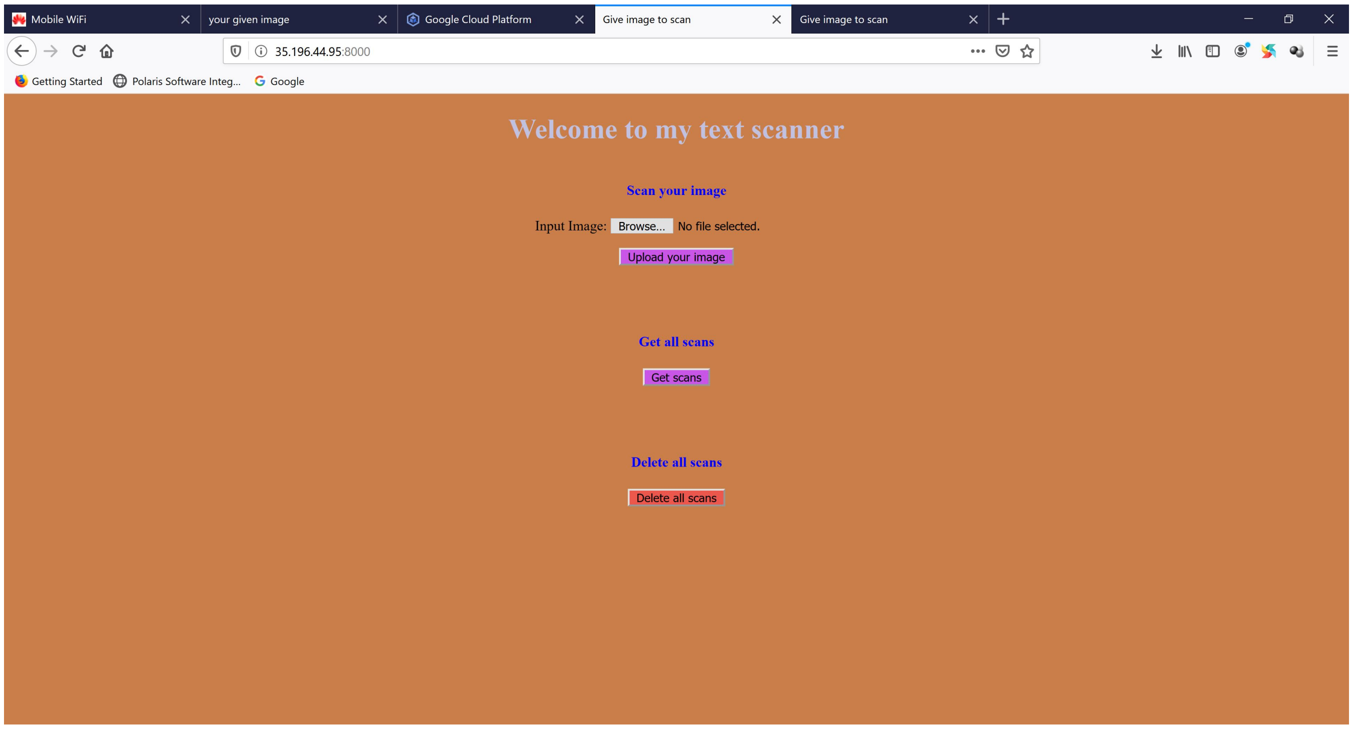


Figure 3.7: Text Scanner over the internet

Bibliography

[1] OpenCV Intel Corporation, Willow Garage, Itseez *OpenCV*.
Initial Relese:June 2000
Stable release 4.3.0 / 3 April 2020
`https://opencv.org`

[2] Django Adrian Holovaty, Simon Willison *Django Software Foundation*
Initial release:21 July 2005
Stable release:3.0.6 / 4 May 2020;
`https://www.djangoproject.com/`

[3] HTML *Refsnes Data*
`https://www.w3schools.com/`

[4] Docker Solomon Hykes *Docker, Inc.*
Stable release:19.03.8 / March 10, 2020
`https://www.docker.com/`

[5] Kuberenetes *Cloud Native Computing Foundation*
Initial release:7 June 2014
Stable release:1.18 / March 25, 2020
`https://kubernetes.io/`