# 

# 图片处理GKE部署文档

目录

[一 概述 2](#_Toc95307030)

[二 部署需要的配置文件 2](#_Toc95307031)

[三 部署步骤 41](#_Toc95307032)

[3.1 编译安装 41](#_Toc95307033)

[3.2 使用镜像包部署 42](#_Toc95307034)

[四 验证 43](#_Toc95307035)

## 一 概述

客户需要一套图片处理程序，通过gke方式提供服务，以达到通过将图片传送到GCS，经过程序的处理，将图片转换成客户想要的格式，本篇文章概述具体的部署方法

## 二 部署需要的配置文件

1. **Dockerfile**

FROM python:3.7

RUN apt-get update && apt-get install -y python3 python3-pip git

# Copy local code to the container image.

ENV APP\_HOME /app

WORKDIR $APP\_HOME

COPY . ./

RUN curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py && \

python3 get-pip.py

RUN pip install Flask gunicorn

RUN pip install Werkzeug cloudstorage google-cloud-datastore Pillow google-cloud-storage google-cloud-pubsub

CMD exec gunicorn --bind :$PORT --workers 1 --threads 8 app:app

1. **app.yaml**

import os

# import pyvips

import re

import mimetypes

from flask import Flask, request, make\_response, send\_file, Response

from PIL import Image, ImageDraw, ImageSequence

from werkzeug.routing import BaseConverter

from google.cloud import storage

from PIL import ImageFile

ImageFile.LOAD\_TRUNCATED\_IMAGES = True

app = Flask(\_\_name\_\_)

IMAGE\_INFO = "imageInfo"

IMAGE\_VIEW = "imageView2"

EXIF = "exif"

IMAGE\_MOGR = "imageMogr2"

WATER\_MARK = "watermark"

IMAGE\_AVE = "imageAve"

def item\_index(arr, item):

"""

获取元素在列表中的索引

:param arr:

:param item:

:return:

"""

for i, value in enumerate(arr):

if value == item:

return i

return

def merge\_dict(source, target):

"""

合并两个字典。合并后的字典的value是一个列表。

eg:

doog\_1 = {name: 'wangwang', age: 10}

doog\_2 = {name: 'wang~', gender: '♂'}

合并以后

doog = {name: ['wangwang', 'wang~'], age: 10, gender: '♂'}

:param source:

:param target:

:return:

"""

# keys = [key for key in source]

keys = list(source)[:]

keys += [key for key in target if not key in keys]

for key in keys:

v1 = source.get(key, [])

v2 = target.get(key, [])

if not isinstance(v1, list):

v1 = [v1]

if isinstance(v2, list):

v1 += v2

else:

v1.append(v2)

source[key] = v1

return source

def parse\_qs(query):

if not query:

return

encoded = {}

args = query.split("/")

interface = args[0]

if IMAGE\_INFO == interface:

encoded["interface"] = IMAGE\_INFO

elif IMAGE\_VIEW == interface:

if len(args) <= 2:

return

encoded["interface"] = IMAGE\_VIEW

encoded["mode"] = args[1]

# ["w", 2, "h", 2] ==> {"w": 2, "h": 2}

params = dict(zip(\*2 \* (iter(args[2:]),)))

merge\_dict(encoded, params)

elif EXIF == interface:

encoded["interface"] = EXIF

elif IMAGE\_MOGR == interface:

encoded["interface"] = IMAGE\_MOGR

encoded["auto-orient"] = str("auto-orient" in args)

encoded["strip"] = str("strip" in args)

encoded["blur"] = str("blur" in args)

args\_name = ["thumbnail", "gravity", "crop", "rotate", "format", "interlace"]

for arg\_name in args\_name:

if arg\_name in args:

try:

encoded[arg\_name] = args[item\_index(args, arg\_name) + 1]

except IndexError:

pass

except TypeError:

pass # NoneType

elif WATER\_MARK == interface:

if len(args) <= 2:

return

encoded["interface"] = WATER\_MARK

encoded["mode"] = args[1]

params = dict(zip(\*2 \* (iter(args[2:]),)))

merge\_dict(encoded, params)

elif IMAGE\_AVE == interface:

encoded["interface"] = IMAGE\_AVE

else:

return

return encoded

def image\_view\_mode\_1(im, w, h):

"""

限定缩略图的宽最少为<Width>，高最少为<Height>，进行等比缩放，居中裁剪。

转后的缩略图通常恰好是 <Width>x<Height> 的大小（有一个边缩放的时候会因为超出矩形框而被裁剪掉多余部分）。

如果只指定 w 参数或只指定 h 参数，代表限定为长宽相等的正方图。

"""

if not w and not h:

return

size = im.size

if not w:

h = int(h)

w = min(h, size[0])

if not h:

w = int(w)

h = min(w, size[1])

w = int(w)

h = int(h)

ratio\_w = w / size[0]

ratio\_h = h / size[1]

max\_ratio = max(ratio\_w, ratio\_h)

min\_ratio = min(ratio\_w, ratio\_h)

if min\_ratio >= 1: # 两边都大

return im

if max\_ratio < 1: # 两边均小于原来

# 新规格

size = resize = tuple(int(x \* max\_ratio) for x in size)

im = im.resize(resize)

box = []

box.append(int((size[0] - w) / 2))

box.append(int((size[1] - h) / 2))

box.append(w + box[0])

box.append(h + box[1])

im = im.crop(tuple(box))

return im

def image\_view\_mode\_2(im, w, h):

"""

限定缩略图的宽最多为<Width>，高最多为<Height>，进行等比缩放，不裁剪。

如果只指定 w 参数则表示限定宽度（高度自适应），只指定 h 参数则表示限定高度（宽度自适应）。

它和模式0类似，区别只是限定宽和高，不是限定长边和短边。

从应用场景来说，模式0适合移动设备上做缩略图，模式2适合PC上做缩略图。

eg:

"""

if not w and not h:

return

size = im.size

ratio\_w = ratio\_h = 1

if w:

w = int(w)

ratio\_w = w / size[0]

if h:

h = int(h)

ratio\_h = h / size[1]

min\_ratio = min(ratio\_w, ratio\_h)

if min\_ratio >= 1:

return im

resize = tuple(int(x \* min\_ratio) for x in size)

im = im.resize(resize)

return im

def image\_view\_mode\_3(im, w, h):

"""

限定缩略图的宽最少为<Width>，高最少为<Height>，进行等比缩放，不裁剪。

"""

if not w and not h:

return

size = im.size

if not w:

w = h

if not h:

h = w

w = int(w)

h = int(h)

ratio\_w = w / size[0]

ratio\_h = h / size[1]

max\_ratio = max(ratio\_w, ratio\_h)

if max\_ratio >= 1:

return im

resize = tuple(int(x \* max\_ratio) for x in size)

im = im.resize(resize)

return im

def image\_view\_mode\_4(im, long\_edge, short\_edge):

"""

限定缩略图的长边最少为<LongEdge>，短边最少为<ShortEdge>，进行等比缩放，不裁剪。

这个模式很适合在手持设备做图片的全屏查看（把这里的长边短边分别设为手机屏幕的分辨率即可），

生成的图片尺寸刚好充满整个屏幕（某一个边可能会超出屏幕）。

"""

if not long\_edge and not short\_edge:

return

size = im.size

origin\_long\_edge = max(size)

origin\_short\_edge = min(size)

if not long\_edge:

long\_edge = short\_edge

if not short\_edge:

short\_edge = long\_edge

long\_edge = int(long\_edge)

short\_edge = int(short\_edge)

ratio\_long = long\_edge / origin\_long\_edge

ratio\_short = short\_edge / origin\_short\_edge

max\_ratio = max(ratio\_long, ratio\_short)

if max\_ratio >= 1:

return im

resize = tuple(int(x \* max\_ratio) for x in size)

im = im.resize(resize)

return im

def image\_view\_mode\_5(im, long\_edge, short\_edge):

"""

限定缩略图的长边最少为<LongEdge>，短边最少为<ShortEdge>，进行等比缩放，居中裁剪。

同上模式4，但超出限定的矩形部分会被裁剪。

"""

if not long\_edge and not short\_edge:

return

size = im.size

origin\_long\_edge = max(size)

origin\_short\_edge = min(size)

if not long\_edge:

short\_edge = int(short\_edge)

long\_edge = short\_edge

if not short\_edge:

long\_edge = int(long\_edge)

short\_edge = long\_edge

long\_edge = min(int(long\_edge), origin\_long\_edge)

short\_edge = min(int(short\_edge), origin\_short\_edge)

ratio\_long = long\_edge / origin\_long\_edge

ratio\_short = short\_edge / origin\_short\_edge

min\_ratio = min(ratio\_long, ratio\_short)

max\_ratio = max(ratio\_long, ratio\_short)

if min\_ratio >= 1:

return im

box = []

if max\_ratio < 1:

size = resize = tuple(int(x \* max\_ratio) for x in size)

im = im.resize(resize)

if size[0] >= size[1]: # 横向

box.append(int((size[0] - long\_edge) / 2))

box.append(int((size[1] - short\_edge) / 2))

box.append(box[0] + long\_edge)

box.append(box[1] + short\_edge)

else: # 竖向

box.append(int((size[0] - short\_edge) / 2))

box.append(int((size[1] - long\_edge) / 2))

box.append(box[0] + short\_edge)

box.append(box[1] + long\_edge)

im = im.crop(tuple(box))

return im

def image\_view\_mode\_6(ima, r, type\_):

size = ima.size

r2 = int(min(size[0], size[1]))

if int(r) <= int(r2 / 2):

r3 = int(r)

else:

r3 = int(r2 / 2)

if size[0] != size[1]:

ima = image\_view\_mode\_1(ima, int(r2), int(r2))

if type\_.lower() == 'jpg' or type\_.lower() == 'jpeg':

imb = Image.new('RGBA', (r3 \* 2, r3 \* 2), (255, 255, 255, 0))

pima = ima.load() # 像素的访问对象

pimb = imb.load()

r = float(r2 / 2) # 圆心横坐标

for i in range(r2):

for j in range(r2):

lx = abs(i - r) # 到圆心距离的横坐标

ly = abs(j - r) # 到圆心距离的纵坐标

l = (pow(lx, 2) + pow(ly, 2)) \*\* 0.5 # 三角函数 半径

if l < r3:

pimb[i - (r - r3), j - (r - r3)] = pima[i, j]

imb = imb.convert('RGB')

return imb

else:

circle = Image.new('L', (r3 \* 2, r3 \* 2), 0)

ima = ima.convert("RGBA")

w, h = ima.size

# draw = ImageDraw.Draw(circle, [(int(w/2-r3), int(h/2-r3))])

draw = ImageDraw.Draw(circle)

draw.ellipse((0, 0, r3 \* 2, r3 \* 2), fill=255)

imb = Image.new('L', ima.size, 255)

imb.paste(circle, (int(w/2-r3), int(h/2-r3)))

ima.putalpha(imb)

crop = str(r3 \* 2) + 'x' + str(r3 \* 2) + 'a' + str(int((r2 - (r3 \* 2)) /2)) + 'a' + str(int((r2 - (r3 \* 2)) /2))

ima = image\_mogr\_crop(ima, '', crop)

return ima

def get\_box(size, point, width, height, dx=0, dy=0):

"""

先趋于中心，后偏移。但是始终在原图范围内

:param size: 数组size[0]底层背景的宽，size[1]底层背景的高

:param point: 中心圆点坐标，左上角为0,0，右下角为size[0],size[1]

:param width: 绿色图层的宽

:param height: 绿色图层的高

:param dx: 向右偏移量

:param dy: 向下偏移量

:return:

"""

width = min(size[0], width)

height = min(size[1], height)

box = [int(point[0] - width / 2), int(point[1] - height / 2), int(point[0] + width / 2), int(point[1] + height / 2)]

if box[0] < 0:

# 先给box[2]赋值，它依赖于box[0]

box[2] -= box[0]

box[0] = 0

if box[1] < 0:

box[3] -= box[1]

box[1] = 0

# 因为width和height永远小于等于外层box的宽和高，上下两种情况不会同时出现

# box[0] < 0 和 box[2] > size[0]不会同时存在

if box[2] > size[0]:

box[0] -= (box[2] - size[0])

box[2] = size[0]

if box[3] > size[1]:

box[1] -= (box[3] - size[1])

box[3] = size[1]

# 首先判断偏移后是否超出原图范围，如果超出则尽最大可能偏移。保证截图仍在原图内

if box[2] + dx > size[0]:

box[0] += (size[0] - box[2])

box[2] = size[0]

else:

box[0] += dx

box[2] += dx

if box[3] + dy > size[1]:

box[1] += (size[1] - box[3])

box[3] = size[1]

else:

box[1] += dy

box[3] += dy

return tuple(box)

def \_get\_gravity\_point(size, gravity):

point = [0, 0]

if "northwest" == gravity:

point[0] = 0

point[1] = 0

elif "north" == gravity:

point[0] = int(size[0] / 3)

point[1] = 0

elif "northeast" == gravity:

point[0] = int(2 \* (size[0] / 3))

point[1] = 0

elif "west" == gravity:

point[0] = 0

point[1] = int(size[1] / 3)

elif "center" == gravity:

point[0] = int(size[0] / 3)

point[1] = int(size[1] / 3)

elif "east" == gravity:

point[0] = int(2 \* (size[0] / 3))

point[1] = int(size[1] / 3)

elif "southwest" == gravity:

point[0] = 0

point[1] = int(2 \* (size[1] / 3))

elif "south" == gravity:

point[0] = int(size[0] / 3)

point[1] = int(2 \* (size[1] / 3))

elif "southeast" == gravity:

point[0] = int(2 \* (size[0] / 3))

point[1] = int(2 \* (size[1] / 3))

return point

def image\_mogr\_crop(im, gravity, crop):

"""

图片裁剪

"""

size = im.size

if gravity:

gravity = gravity.lower()

point = \_get\_gravity\_point(size, gravity)

if re.match(r"^([1-9][0-9]\*)x$", crop):

width = int(crop[:-1])

if width >= 10000:

return im

box = get\_box(size, point, width, size[1])

im = im.crop(box)

elif re.match(r"^x([1-9][0-9]\*)$", crop):

height = int(crop[1:])

if height >= 10000:

return im

box = get\_box(size, point, size[0], height)

im = im.crop(box)

elif re.match(r"^([1-9][0-9]\*)x([1-9][0-9]\*)$", crop):

crop = [int(x) for x in crop.split("x")]

if min(crop) >= 10000:

return im

box = get\_box(size, point, crop[0], crop[1])

im = im.crop(box)

# elif re.match(r"^([1-9][0-9]\*)x([1-9][0-9]\*)a([1-9][0-9]\*)a([1-9][0-9]\*)$", crop):

elif re.match(r"^([1-9][0-9]\*)x([1-9][0-9]\*)a([0-9][0-9]\*)a([0-9][0-9]\*)$", crop):

# /crop/{cropSize}a<dx>a<dy>

# 相对于偏移锚点，向右偏移dx个像素，同时向下偏移dy个像素。

crop = [int(x) for x in re.findall(r"[0-9][0-9]\*", crop)]

if min(crop[:2]) >= 10000:

return im

# point[0] += crop[2]

# point[1] += crop[3]

box = get\_box(size, point, crop[0], crop[1], crop[2], crop[3])

im = im.crop(box)

return im

def image\_mogr\_auto\_orient(im):

"""

根据原图EXIF信息自动旋正，便于后续处理建议放在首位。

1 2 3 4 5 6 7 8

888888 888888 88 88 8888888888 88 88 8888888888

88 88 88 88 88 88 88 88 88 88 88 88

8888 8888 8888 8888 88 8888888888 8888888888 88

88 88 88 88

88 88 888888 888888

:rtype : Image

:param im:

"""

print('auto\_orient on')

try:

exif = im.\_getexif()

except:

return im

if exif and exif.get(0x0112, None):

orientation = exif.get(0x0112, None)

if orientation == 1:

pass

elif orientation == 2:

im = im.transpose(Image.FLIP\_LEFT\_RIGHT)

elif orientation == 3:

im = im.transpose(Image.ROTATE\_180)

elif orientation == 4:

im = im.transpose(Image.FLIP\_TOP\_BOTTOM)

elif orientation == 5:

im = im.transpose(Image.ROTATE\_270).transpose(Image.FLIP\_LEFT\_RIGHT)

elif orientation == 6:

im = im.transpose(Image.ROTATE\_270)

elif orientation == 7:

im = im.transpose(Image.ROTATE\_90).transpose(Image.FLIP\_LEFT\_RIGHT)

elif orientation == 8:

im = im.transpose(Image.ROTATE\_90)

# 重新修正Orientation值

# im['Orientation'] = 1

return im

def file\_to\_binary(p, type\_=None):

if not type\_:

suffix = re.findall(r'\.[^.\\/:\*?"<>|\r\n]+$', p)[0][1:]

type\_ = suffix.lower()

if 'Range' in request.headers:

start, end = get\_range(request)

response = partial\_response(p, start, end)

else:

response = make\_response(send\_file(p, conditional=True))

response.headers['Content-Type'] = 'image' + '/' + str(type\_)

response.headers['Content-Disposition'] = 'inline'

response.headers['Accept-Ranges'] = 'bytes'

response.cache\_control.max\_age = 86400

response.cache\_control.public = True

return response

def partial\_response(path, start, end=None):

file\_size = os.path.getsize(path)

if end is None:

end = file\_size - start - 1

end = min(end, file\_size - 1)

length = end - start + 1

with open(path, 'rb') as fd:

fd.seek(start)

bytes = fd.read(length)

response = Response(

bytes,

206, # Partial Content

mimetype=mimetypes.guess\_type(path)[0], # Content-Type must be correct

direct\_passthrough=True, # Identity encoding

)

response.headers.add(

'Content-Range', 'bytes {0}-{1}/{2}'.format(

start, end, file\_size,

),

)

return response

def get\_range(request):

range = request.headers.get('Range')

m = re.match('bytes=(?P<start>\d+)-(?P<end>\d+)?', range)

if m:

start = m.group('start')

end = m.group('end')

start = int(start)

if end is not None:

end = int(end)

return start, end

else:

return 0, None

# 处理缩略图

def thumbnail\_do(file\_name, size\_w, size\_h):

key = os.getcwd() + '/' + file\_name

im = Image.open(key)

im.thumbnail((size\_w, size\_h))

file\_k = os.getcwd() + '/' + 'thumbnail\_' + file\_name

im.save(file\_k, im.format)

file\_name = 'thumbnail\_' + file\_name

return file\_name

def resize\_do(file\_name, size\_w, size\_h):

key = os.getcwd() + '/' + file\_name

im = Image.open(key)

re = im.resize((size\_w, size\_h))

file\_k = os.getcwd() + '/' + 'thumbnail1\_' + file\_name

re.save(file\_k, im.format)

return file\_k

# 处理裁剪图片

def crop\_do(file\_name, left, top, right, bottom):

key = os.getcwd() + '/' + file\_name

im = Image.open(key)

cr = im.crop((left, top, right, bottom))

file\_k = os.getcwd() + '/' + 'crop2\_' + file\_name

cr.save(file\_k, im.format)

file\_name = 'crop2\_' + file\_name

return file\_name

# 处理格式转换

def convert\_do(file\_name, type\_):

if type\_ == 'jpg':

type\_ = 'jpeg'

key = os.getcwd() + '/' + file\_name

im = Image.open(key)

suffix = re.findall(r'\.[^.\\/:\*?"<>|\r\n]+$', file\_name)[0][1:]

file\_k = os.getcwd() + '/' + 'convert3\_' + file\_name.split(suffix)[0] + type\_

im.save(file\_k, type\_)

file\_name = 'convert3\_' + file\_name.split(suffix)[0] + type\_

return file\_name

def toheic(filename):

i = pyvips.Image.new\_from\_file(filename)

suffix = re.findall(r'\.[^.\\/:\*?"<>|\r\n]+$', filename)[0][1:]

file\_name = filename.split(suffix)[0] + 'heic'

i.write\_to\_file(file\_name)

return file\_name

def download\_blob(bucket\_name, source\_blob\_name):

"""Downloads a blob from the bucket."""

file\_name = re.split('/', source\_blob\_name)[-1]

destination\_file\_name = os.getcwd() + '/' + file\_name

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

blob = bucket.blob(source\_blob\_name)

blob.download\_to\_filename(destination\_file\_name)

print('Blob {} downloaded to {}.'.format(

source\_blob\_name,

destination\_file\_name))

@app.route('/', methods=["GET", "POST"])

def hello():

return 'index'

class RegexConverter(BaseConverter):

def \_\_init\_\_(self, url\_map, \*args):

super(RegexConverter, self).\_\_init\_\_(url\_map)

self.regex = args[0]

app.url\_map.converters['re'] = RegexConverter

@app.route('/<re(r"[\w\W]\*"):route\_file>', methods=['GET', 'POST'])

def image2(route\_file):

request\_file = re.split('/', route\_file)[-1]

request\_action = request.args.get("x-oss-process")

bucket\_name = os.getenv('BUCKET\_NAME')

download\_blob(bucket\_name, route\_file)

suffix = re.findall(r'\.[^.\\/:\*?"<>|\r\n]+$', request\_file)[0][1:]

type\_ = suffix

quality = 75

if not request\_action:

return file\_to\_binary(request\_file, suffix)

key = os.getcwd() + '/' + request\_file

if type\_.lower() == 'gif':

gif = Image.open(key)

dura = gif.info['duration']

imgs = [f.copy() for f in ImageSequence.Iterator(gif)]

index = 0

imglist = []

os.mkdir("imagesttt")

for frame in imgs:

frame.save("./imagesttt/%d.png" % index)

im = Image.open("./imagesttt/%d.png" % index)

if re.findall('auto-orient', request\_action):

im = image\_mogr\_auto\_orient(im)

req = request\_action.split('/')

if req[0] == 'image':

for i in req:

if re.findall('format', i):

act\_for = request\_action.split('/')[req.index(i)]

act\_for = act\_for.split(',')

if act\_for[0] == 'format':

type\_ = act\_for[1]

if type\_.lower() == 'jpg':

type\_ = 'jpeg'

# if type\_.lower() == 'heic':

# filename = toheic(request\_file)

# file\_k = os.getcwd() + '/' + filename

# return file\_to\_binary(file\_k, 'heic')

for i in req:

if re.findall('quality', i):

act\_for = request\_action.split('/')[req.index(i)]

act\_for = act\_for.split(',')

if act\_for[0] == 'quality':

quality = act\_for[1].split('\_')[1]

if re.findall('\_', i):

act\_1 = req[req.index(i)]

act\_2 = act\_1.split(',')

if act\_2[0] == 'crop':

act\_2.pop(0)

act\_d = {}

for i in act\_2:

act\_d[i.split('\_')[0]] = i.split('\_')[1]

w = act\_d.get('w', str(im.size[0]))

h = act\_d.get('h', str(im.size[1]))

x = act\_d.get('x', '0')

y = act\_d.get('y', '0')

if int(im.size[0]) < (int(w) + int(x)):

w = str(int(im.size[0]) - int(x))

if int(im.size[1]) < (int(h) + int(y)):

h = str(int(im.size[1]) - int(y))

crop = w + 'x' + h + 'a' + x + 'a' + y

g = act\_d.get('g')

if g:

if g == 'nw':

g = 'northwest'

elif g == 'ne':

g = 'northeast'

elif g == 'sw':

g = 'southwest'

elif g == 'se':

g = 'southeast'

im = image\_mogr\_crop(im, g, crop)

elif act\_2[0] == 'resize':

act\_2.pop(0)

act\_d = {}

for i in act\_2:

act\_d[i.split('\_')[0]] = i.split('\_')[1]

m = act\_d.get('m')

if not m:

m = 'lfit'

w = act\_d.get('w', im.size[0])

h = act\_d.get('h', im.size[1])

l = act\_d.get('l')

s = act\_d.get('s')

p = act\_d.get('p')

if l:

im = image\_view\_mode\_2(im, l, l)

if s:

im = image\_view\_mode\_3(im, s, s)

if m == 'lfit':

im = image\_view\_mode\_2(im, w, h)

elif m == 'mfit':

im = image\_view\_mode\_3(im, w, h)

elif m == 'fill':

im = image\_view\_mode\_1(im, w, h)

elif m == 'fixed':

im = im.resize((int(w), int(h)))

else:

return 'm err'

if p:

w = im.size[0] \* (int(p) / 100)

h = im.size[1] \* (int(p) / 100)

im = image\_view\_mode\_2(im, w, h)

elif act\_2[0] == 'circle':

act\_2.pop(0)

act\_d = {}

for i in act\_2:

act\_d[i.split('\_')[0]] = i.split('\_')[1]

r = act\_d.get('r')

if not r:

return 'r err'

im = image\_view\_mode\_6(im, r, type\_)

imglist.append(im)

index += 1

file\_k = os.getcwd() + '/' + str(request\_action).replace('/', '') + '\_' + request\_file

os.system("rm -rf ./imagesttt")

if quality != 75:

imglist[0].save(file\_k, type\_, save\_all=True, append\_images=imglist[1:], loop=0, duration=dura, quality=int(quality))

else:

imglist[0].save(file\_k, type\_, save\_all=True, append\_images=imglist[1:], loop=0, duration=dura)

return file\_to\_binary(file\_k, type\_)

im = Image.open(request\_file)

if re.findall('auto-orient', request\_action):

im = image\_mogr\_auto\_orient(im)

req = request\_action.split('/')

if req[0] == 'image':

for i in req:

if re.findall('format', i):

act\_for = request\_action.split('/')[req.index(i)]

act\_for = act\_for.split(',')

if act\_for[0] == 'format':

type\_ = act\_for[1]

if type\_.lower() == 'jpg':

type\_ = 'jpeg'

# if type\_.lower() == 'heic':

# filename = toheic(request\_file)

# file\_k = os.getcwd() + '/' + filename

# return file\_to\_binary(file\_k, 'heic')

for i in req:

if re.findall('quality', i):

act\_for = request\_action.split('/')[req.index(i)]

act\_for = act\_for.split(',')

if act\_for[0] == 'quality':

quality = act\_for[1].split('\_')[1]

if re.findall('\_', i):

act\_1 = req[req.index(i)]

act\_2 = act\_1.split(',')

if act\_2[0] == 'crop':

act\_2.pop(0)

act\_d = {}

for i in act\_2:

act\_d[i.split('\_')[0]] = i.split('\_')[1]

w = act\_d.get('w', str(im.size[0]))

h = act\_d.get('h', str(im.size[1]))

x = act\_d.get('x', '0')

y = act\_d.get('y', '0')

if int(im.size[0]) < (int(w) + int(x)):

w = str(int(im.size[0]) - int(x))

if int(im.size[1]) < (int(h) + int(y)):

h = str(int(im.size[1]) - int(y))

crop = w + 'x' + h + 'a' + x + 'a' + y

g = act\_d.get('g')

if g:

if g == 'nw':

g = 'northwest'

elif g == 'ne':

g = 'northeast'

elif g == 'sw':

g = 'southwest'

elif g == 'se':

g = 'southeast'

im = image\_mogr\_crop(im, g, crop)

elif act\_2[0] == 'resize':

act\_2.pop(0)

act\_d = {}

for i in act\_2:

act\_d[i.split('\_')[0]] = i.split('\_')[1]

m = act\_d.get('m')

if not m:

m = 'lfit'

w = act\_d.get('w', im.size[0])

h = act\_d.get('h', im.size[1])

l = act\_d.get('l')

s = act\_d.get('s')

p = act\_d.get('p')

if l:

im = image\_view\_mode\_2(im, l, l)

if s:

im = image\_view\_mode\_3(im, s, s)

if m == 'lfit':

im = image\_view\_mode\_2(im, w, h)

elif m == 'mfit':

im = image\_view\_mode\_3(im, w, h)

elif m == 'fill':

im = image\_view\_mode\_1(im, w, h)

elif m == 'fixed':

im = im.resize((int(w), int(h)))

else:

return 'm err'

if p:

w = im.size[0] \* (int(p) / 100)

h = im.size[1] \* (int(p) / 100)

im = image\_view\_mode\_2(im, w, h)

elif act\_2[0] == 'circle':

act\_2.pop(0)

act\_d = {}

for i in act\_2:

act\_d[i.split('\_')[0]] = i.split('\_')[1]

r = act\_d.get('r')

if not r:

return 'r err'

im = image\_view\_mode\_6(im, r, type\_)

file\_k = os.getcwd() + '/' + str(request\_action).replace('/', '') + '\_' + request\_file

if type\_.lower() == 'jpg':

type\_ = 'jpeg'

if suffix.lower() == 'jpg':

suffix = 'jpeg'

if type\_.lower() == 'heic' or type\_.lower() == 'heif':

im.save(file\_k, suffix)

filename = toheic(file\_k)

return file\_to\_binary(filename, 'heic')

if quality != 75:

im.save(file\_k, type\_, quality=int(quality))

else:

im.save(file\_k, type\_)

return file\_to\_binary(file\_k, type\_)

# if request\_action == 'thumbnail':

# size\_w = request.args.get('size\_w')

# size\_h = request.args.get('size\_h')

# if request\_file and size\_w and size\_h:

# download\_blob(bucket\_name, request\_file)

# request\_file = resize\_do(request\_file, int(size\_w), int(size\_h))

# else:

# raise 'miss parameter1'

# elif request\_action == 'crop':

# left = request.args.get('left')

# top = request.args.get('top')

# right = request.args.get('right')

# bottom = request.args.get('bottom')

# if request\_file and left and top and right and bottom:

# download\_blob(bucket\_name, request\_file)

# request\_file = crop\_do(request\_file, int(left), int(top), int(right), int(bottom))

# else:

# raise 'miss parameter2'

# elif request\_action == 'convert':

# type\_ = request.args.get("type")

# if type\_ and request\_file:

# download\_blob(bucket\_name, request\_file)

# if type\_ == 'heic':

# request\_file = toheic(request\_file)

# else:

# request\_file = convert\_do(request\_file, type\_)

# try:

# return file\_to\_binary(request\_file, type\_)

# except:

# return 'miss parameter'

if \_\_name\_\_ == '\_\_main\_\_':

# HOST = '0.0.0.0'

# PORT = 8080

# app.run(HOST, PORT, debug=True)

app.run()

1. **backend-config.yaml**

apiVersion: cloud.google.com/v1beta1

kind: BackendConfig

metadata:

name: backend-config

spec:

cdn:

enabled: true

cachePolicy:

includeHost: true

includeProtocol: true

includeQueryString: true

1. **deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: image-flask

spec:

replicas: 1

selector:

matchLabels:

app: image

template:

metadata:

labels:

app: image

spec:

containers:

- name: image-function

image: us-central1-docker.pkg.dev/bosicloud-testing/image-repo/image-gke:latest

ports:

- containerPort: 8080

resources:

requests:

cpu: 200m

env:

- name: PORT

value: "8080"

- name: BUCKET\_NAME

value: "image-test-test"

1. **ingress.yaml**

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: image-ingress

annotations:

kubernetes.io/ingress.global-static-ip-name: "image-function-address"

spec:

rules:

- http:

paths:

- path: /\*

backend:

serviceName: image

servicePort: 80

1. **service.yaml**

apiVersion: v1

kind: Service

metadata:

name: image

labels:

app: image

annotations:

beta.cloud.google.com/backend-config: '{"ports": {"80":"backend-config"}}'

spec:

type: NodePort

selector:

app: image

ports:

- protocol: TCP

port: 80

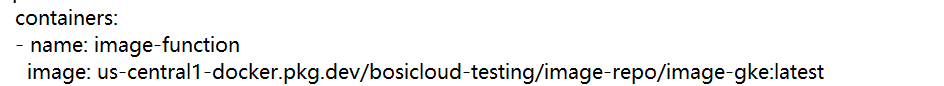
targetPort: 8080

## 三 部署步骤

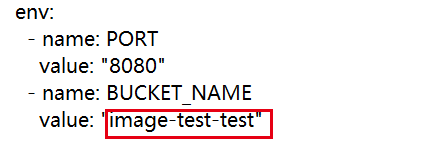
## 3.1 编译安装

正情况是通过这种情况安装部署。

1. 将image-gke.rar上传到服务器，执行 mv image-gke.rar /home && unzip /home/image-gke.rar，如果解压不了，在本地解压后再上传
2. Cd /home/image-gke/docker\_
3. Docker build –t gcr.io/项目名/包名 .
4. Gcloud docker -- push gcr.io/项目名/包名
5. Sudo vim/home/image-gke/gke/deployment.yaml,修改image的值改成gcr.io/项目名/包名格式为： image：gcr.io/项目名/包名



将bucket修改为自己的bucket



修改完成后保存

1. Cd /home/image-gke/gke/ 执行

kubectl apply –f backend-config.yaml

kubectl apply –f deployment.yaml

kubectl apply –f ingress.yaml

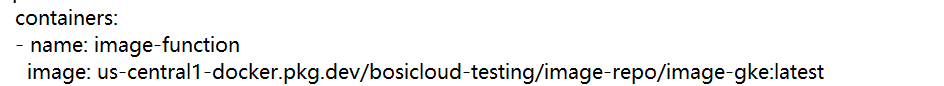
kubectl apply –f service.yaml

1. 查看结果：kubectl get pod，看到状态为Runing即为部署成功

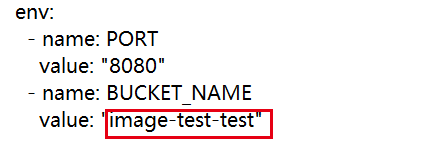
## 3.2 使用镜像包部署

正情况是通过这种情况安装部署。

1. 将image-gke.rar上传到服务器，执行 mv image-gke.rar /home && unzip /home/image-gke.rar，如果解压不了，在本地解压后再上传
2. Cd /home/image-gke/docker\_
3. docker load -i gcr.io.bosicloud-testing.image-fnction-1.tar.gz
4. docker tag gcr.io/bosicloud-testing/image-fnction-1 gcr.io/项目名/包名
5. Gcloud docker -- push gcr.io/项目名/包名
6. Sudo vim/home/image-gke/gke/deployment.yaml,修改image的值改成gcr.io/项目名/包名格式为： image：gcr.io/项目名/包名



将bucket修改为自己的bucket



修改完成后保存

1. Cd /home/image-gke/gke/ 执行

kubectl apply –f backend-config.yaml

kubectl apply –f deployment.yaml

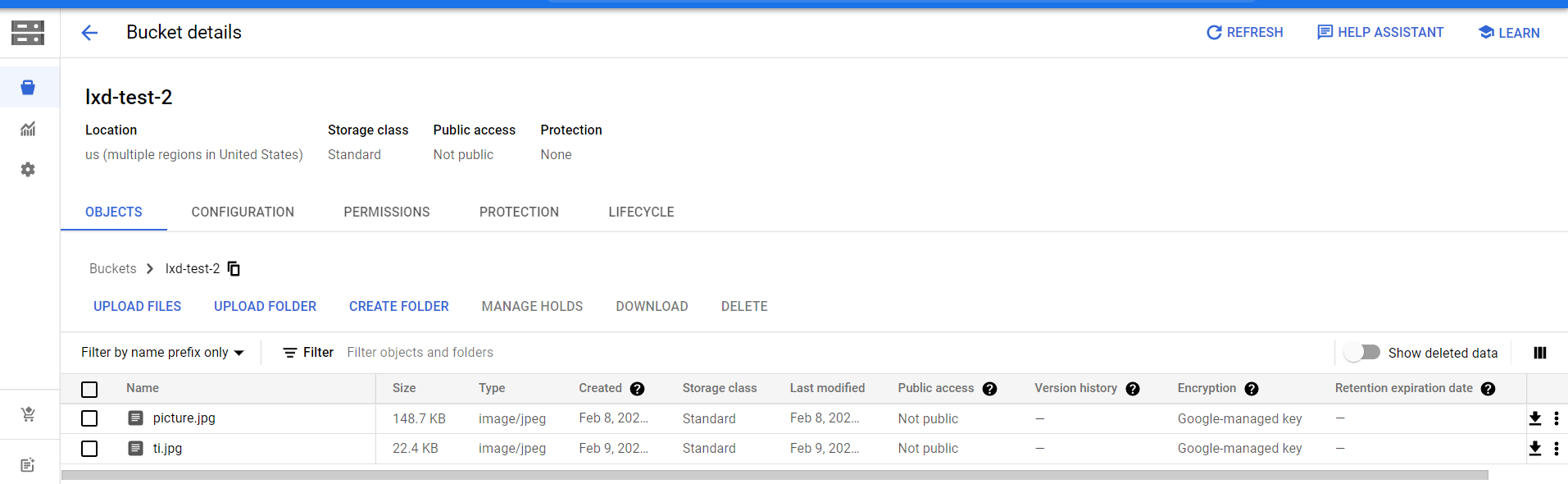
kubectl apply –f ingress.yaml

kubectl apply –f service.yaml

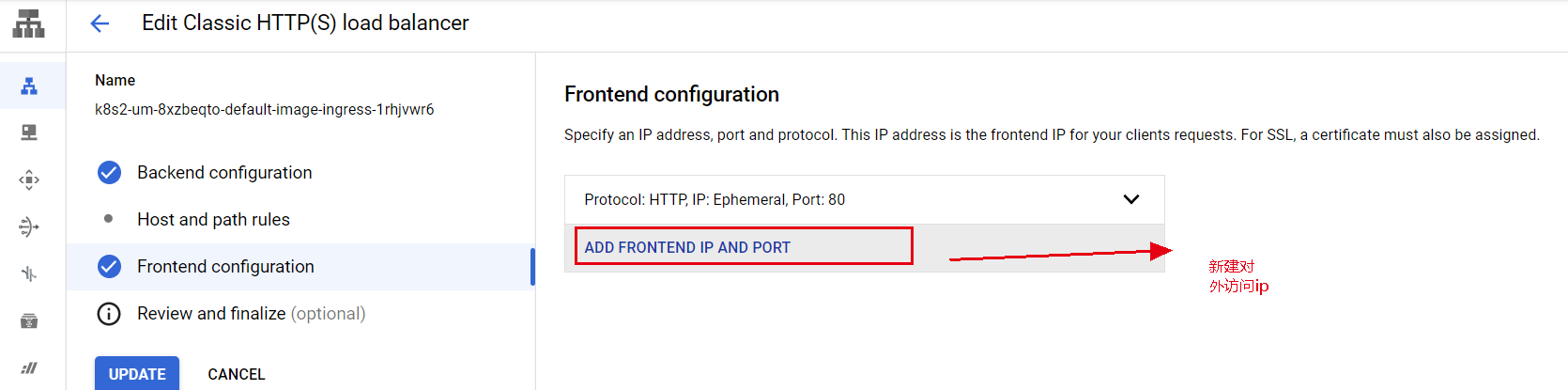
1. 查看结果：kubectl get pod，看到状态为Runing即为部署成功

## 四 验证

打开GCS创建以deployment.Yaml修改的bucket名字为名，然后上传测试照片

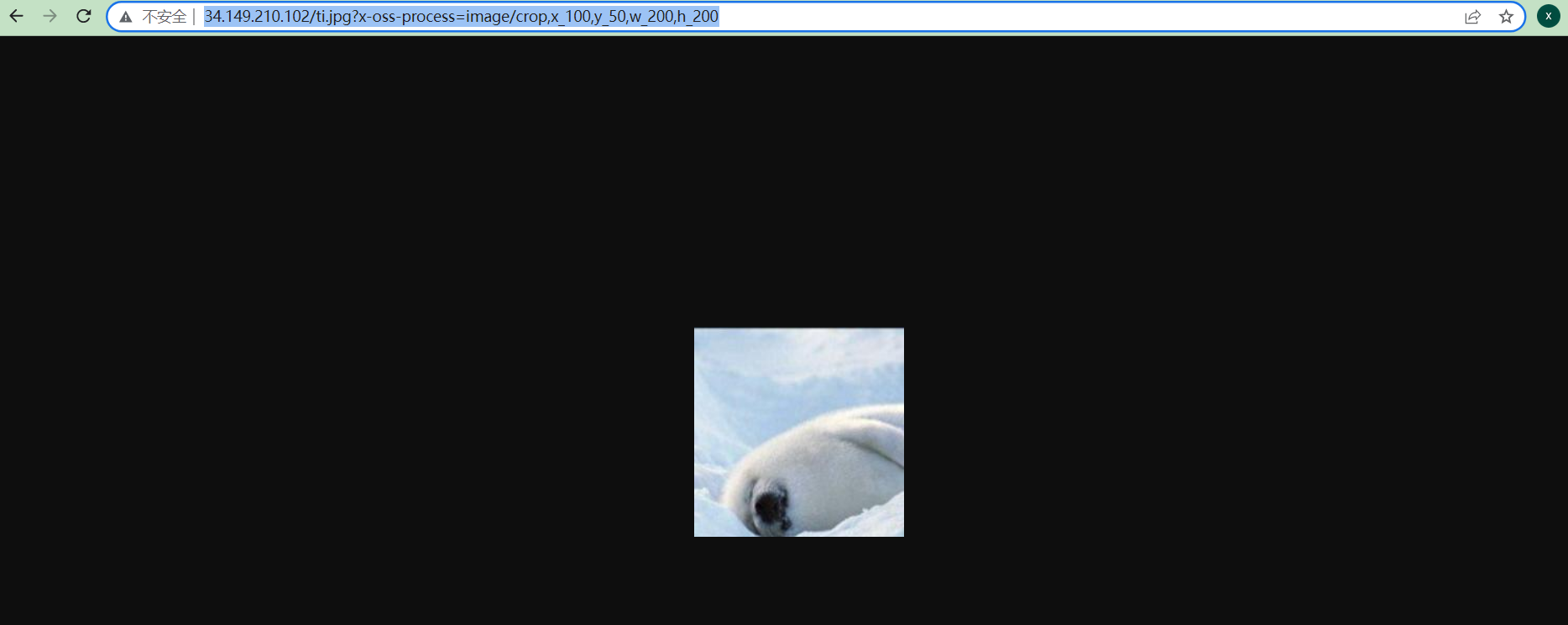


打开glb找到集群创建的glb，如果没有对外IP的，点击编辑，自己点击新建一个



<http://34.149.210.102/ti.jpg?x-oss-process=image/crop,x_100,y_50,w_200,h_200>

结果如下：



参数访问出现这个效果，部署成功。测试文档请参考《《图片处理功能》》