| GROUP3 MEMBERS MESUT BUNALDI  MUHAMMED KANDUR  HİKMET TÜTÜNCÜ  ONGUN ALP BABA MUHAMMET BURAK ÖZÇELİK  LEMYE CEREN GÜMÜŞ  GÖKTUĞ ALİ AKIN  FERDİ SÖNMEZ  OSMAN ÇETİN  EMRE DURAK  MELİKE SERRA KALYON HASNAIN ALI | | DEPLOY3 JANUARY 2019 | | --- |  REPORT 7 The group member who did not attend the meeting is MUHAMMED KANDUR,GOKTUG ALI AKIN,EMRE DURAK,HASNAIN ALI  from flask import Flask, jsonify, request #import objects from the Flask model  app = Flask(\_\_name\_\_) #define app using Flask  languages = [{'name' : 'JavaScript'}, {'name' : 'Python'}, {'name' : 'Ruby'}]  @app.route('/', methods=['GET'])  def test():  return jsonify({'message' : 'It works!'})  @app.route('/lang', methods=['GET'])  def returnAll():  return jsonify({'languages' : languages})  @app.route('/lang/<string:name>', methods=['GET'])  def returnOne(name):  langs = [language for language in languages if language['name'] == name]  return jsonify({'language' : langs[0]})  @app.route('/lang', methods=['POST'])  def addOne():  language = {'name' : request.json['name']}  languages.append(language)  return jsonify({'languages' : languages})  @app.route('/lang/<string:name>', methods=['PUT'])  def editOne(name):  langs = [language for language in languages if language['name'] == name]  langs[0]['name'] = request.json['name']  return jsonify({'language' : langs[0]})  @app.route('/lang/<string:name>', methods=['DELETE'])  def removeOne(name):  lang = [language for language in languages if language['name'] == name]  languages.remove(lang[0])  return jsonify({'languages' : languages})  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True, port=8080) #run app on port 8080 in debug mode  **SAMPLE DEPLOY REQUEST JSON FILE**  {  "Deployment":"Undeploy",  "targetIP":"targetIP",  "targetPswd":"gtu2017A",  "destination":"3",  "source":"3"  }  **SAMPLE UNDEPLOY REQUEST JSON FILE**  {  "Deployment":"Undeploy",  "containerID":"5ad1c5f0b9e8",  "targetIP":"targetIP",  "targetPswd":"gtu2017A",  "destination":"3",  "source":"3"  }  **SAMPLE DEPLOY SUCCESS RESPONSE JSON FILE**  {"Deployment": [{  "State": "Success",  "Deployment": "Deploy",  "ContainerID": "8b254e53a2b8",  "destination": "3",  "source": "3"}]}  **SAMPLE DEPLOY FAIL RESPONSE JSON FILE**  {"Deployment":  [{"State": "Failed",  "Deployment": "Deploy",  "ContainerID": "8b254e53a2b8",  "destination": "3",  "source": "3"}]}  **SAMPLE UNDEPLOY SUCCESS RESPONSE JSON FILE**  {"Deployment": [{  "State": "Success",  "Deployment": "Undeploy",  "ContainerID": "8b254e53a2b8",  "destination": "3",  "source": "3"}]}  **SAMPLE UNDEPLOY FAIL RESPONSE JSON FILE**  {"Deployment":  [{"State": "Failed",  "Deployment": "Undeploy",  "ContainerID": "8b254e53a2b8",  "destination": "3",  "source": "3"}]}  **SENDING JSON RESPONSE**  First to test if we can send a response we need to make a listener.  API Client in flask :  <https://github.com/PrettyPrinted/http_request_python_tutorial>  download the restful.py file.  API is in flask so first download flask.  $ sudo apt install python3-flask  Listens the port. This is needed to test if we can send the response.    When we run our code, sends a response json file to the destination.    UNDEPLOY:  1- Gets request  2- Parses the json file  3- Deploys or Undeploys  4- Sends respons**Anypoint**  HTTP : Listens to port  Logger :  File:  Python:    Through postman send Json file    Http listens to the port if a request comes, the request goes through the logger. Logger keeps the request, file saves the json file to the given path. Then the python code gets the json file from the path and parses it. Deploys or undeploys then sends the response to the destination.  Deployement :  import json  import os  import requests  import json  import subprocess  def jsonRequestParsing():  global containerID,repoURL,targetIP,targetPswd,source,destination  with open('/home/mel/Desktop/software/python/jsonRequest.json', 'r') as f:  distros\_dict = json.load(f)  for distro in distros\_dict:  if(distro == "containerID"):  containerID = distros\_dict[distro]  if(distro == "repoURL"):  repoURL = distros\_dict[distro]  if(distro == "targetIP"):  targetIP = distros\_dict[distro]  if(distro == "targetPswd"):  targetPswd = distros\_dict[distro]  if(distro == "destination"):  destination = distros\_dict[distro]  if(distro == "source"):  source = distros\_dict[distro]    if(distros\_dict["Deployment"]=="Undeploy"):  undeploy()  elif (distros\_dict["Deployment"]=="Deploy"):  deploy()  def undeploy():  print("--------UNDEPLOY--------")  global containerID,repoURL,targetIP,targetPswd,source,destination  sudoPassword = targetPswd  undeployCommand= 'sudo docker rm -f ' + str( containerID)  listRunningContainersCommand= 'sudo docker ps'  print("\t\t\t------------------- \n\t\t\t------- 1 --------- \n\t\t\t------------------- \n")  p=os.popen('echo %s|sudo -S %s' % (sudoPassword, listRunningContainersCommand)).read()  print("\t\t\tRunning Container list:")  print(p)  # Selects State up containers with this command sudo docker ps -a | grep "Up" | cut -d " " -f 1  p=os.popen('echo %s|sudo -S sudo docker ps -a | grep "Up" | cut -d " " -f 1' % (sudoPassword)).read()  print("\t\t\tState Up Container:")  print(p)  print("\t\t\tGiven Container ID:")  print(containerID)  containerlist = p.split('\n')    # Check if there is a container that matches the containerID  flag=0  for i in range(0 , len(containerlist)):  if containerlist[i] == containerID :  flag =1  if flag == 1 :  # There is a container that matches the containerID  print("\t\t\t------------------- \n\t\t\t------- 2 --------- \n\t\t\t------------------- \n")  print("\t\t\tDeleting Container:")  p=os.popen('echo %s|sudo -S %s' % (sudoPassword, undeployCommand)).read()  print(p)  print("\t\t\t------------------- \n\t\t\t------- 3 --------- \n\t\t\t------------------- \n")  a=None  a=os.popen('echo %s|sudo -S sudo docker ps -a | grep "Up" | cut -d " " -f 1' % (sudoPassword)).read()  print("\t\t\tRunning Container List:")  print(a)  #if there isn't any containers running it is succesfull  print("##",a,"##")  if (len(a) <= 2) :  writeToJasonDeployed("success")  sendJsonFile()  else :  writeToJasonDeployed("fail")  sendJsonFile()  else :  # There is no container that matches the containerID  writeToJasonDeployed("fail")  sendJsonFile()  def deploy():  """  TODO  If the container should stay in state up:  to fix the problem open a second terminal to see if there is a container created,  If the container is in state exited no need to open second terminal. Works fine.  """  print("--------DEPLOY--------")  global containerID,repoURL,targetIP,targetPswd,source,destination  """TODO"""  programName = "wordpress"  """TODO"""  sudoPassword=targetPswd  deployCommand= 'sudo docker run '+ str( programName)  listRunningContainersCommand= 'sudo docker ps'    p = os.popen('echo %s|sudo -S %s' % (sudoPassword, listRunningContainersCommand)).read()  print("\t\t\t------------------- \n\t\t\t------- 1 --------- \n\t\t\t------------------- \n")  print("\t\t\tRunning Container list:")  print(p)  # p= os.popen('echo %s|sudo -S %s' % (sudoPassword, deployCommand)).read()  print("\t\t\t------------------- \n\t\t\t------- 2 --------- \n\t\t\t------------------- \n")  print("\t\t\tCreating Container")  print(p)  p=os.popen('echo %s|sudo -S %s' % (sudoPassword, listRunningContainersCommand)).read()  print("\t\t\t------------------- \n\t\t\t------- 3 --------- \n\t\t\t------------------- \n")  print("\t\t\tRunning Container List:")  print(p)  a=None  a=os.popen('echo %s|sudo -S sudo docker ps -a | grep "Up" | cut -d " " -f 1' % (sudoPassword)).read()  print("\t\t\t------------------- \n\t\t\t------- 4 --------- \n\t\t\t------------------- \n")  print("\t\tContainer ID of the Container Status is Up:")  print(a)  #If a isn't null there is a running container.  containerID=None  containerID = a.split('\n')[0]  #change the condition  if (containerID != None) :  writeToJasonDeployed("success")  sendJsonFile()  else :  writeToJasonDeployed("fail")  sendJsonFile()  def writeToJasonDeployed(info):  data = {}  data['Deployment'] = []  if info == "success" :  data['Deployment'].append({  'State': 'Success',  'Deployment':'Deploy',  'ContainerID' : containerID ,  'destination':destination,  'source':source  })  elif info == "fail" :  data['Deployment'].append({  'State': 'Failed',  'Deployment':'Deploy',  'ContainerID' : containerID ,  'destination':destination,  'source':source  })  with open('/home/mel/Desktop/software/python/response.json', 'w') as outfile:  json.dump(data, outfile)  def writeToJasonUndeployed(info):  data = {}  data['Deployment'] = []  if info == "success" :  data['Deployment'].append({  'State': 'Success',  'Deployment':'Undeploy',  'destination':destination,  'source':source  })  elif info == "fail":  data['Deployment'].append({  'State': 'Failed',  'Deployment':'Undeploy',  'destination':destination,  'source':source  })  with open('/home/mel/Desktop/software/python/response.json', 'w') as outfile:  json.dump(data, outfile)  def sendJsonFile():  jsonFile = open(r'/home/mel/Desktop/software/python/response.json', 'r')  data = json.load(jsonFile)  print (data)  # sending post request and saving response as response object  r = requests.post(url = 'http://localhost:8080', data = json.dumps(data))  # extracting response text  sent\_json = r.headers  print("\nResponse:%s"%sent\_json)  containerID = 0  repoURL = ""  targetIP = ""  targetPswd =""  source = 0  destination =0  jsonRequestParsing()  #os.system("gnome-terminal --disable-factory")  """ TODO  container ID = sudo docker ps -a | grep "Up" | cut -d " " -f 1  """  RESOURCES :  <https://blog.oddbit.com/post/2014-08-11-four-ways-to-connect-a-docker/>  <https://nickjanetakis.com/blog/docker-tip-73-connecting-to-a-remote-docker-daemon>  <https://github.com/gokhanhas/my-app>  <https://cloudkul.com/blog/understanding-communication-docker-containers/>  DEPLOY |
| --- | --- | --- |