**Cloud Computing Homework 4**

Improve "to-do list" application's design by using web service. To be more specific, leave the application front-end and overall structure unchanged, but provide the data service through a typical RESTful API instead of supporting directly by a database as in the solution posted for the previous homework.

For example, the original code for root URL

@app.route("/")

def show\_list():

db = get\_db()

cur = db.execute('SELECT what\_to\_do, due\_date, status FROM entries')

entries = cur.fetchall()

tdlist = [dict(what\_to\_do=row[0], due\_date=row[1], status=row[2])

for row in entries]

return render\_template('index.html', todolist=tdlist)

will become

@app.route("/")

def show\_list():

with urllib.request.urlopen('http://localhost:6000/api/items') as response:

resp = response.read()

resp = json.loads(resp)

return render\_template('index.html', todolist=resp)

, assuming the servce runs on port 6000.

Of course, you also need to implement the service itself, which will be similar in structure to the original app, except returning JSON instead of HTML. The corresponding function would be like this

@app.route("/api/items")

def get\_items():

db = get\_db()

cur = db.execute('SELECT what\_to\_do, due\_date, status FROM entries')

entries = cur.fetchall()

tdlist = [dict(what\_to\_do=row[0], due\_date=row[1], status=row[2])

for row in entries]

return jsonify(tdlist)

1. Using Docker container technology and kubernetes, deploy the 'to-do list' app to a cluster on Google Cloud Platform. You need to describe how you create the Docker image, e.g. using a *Dockerfile*, and how you create the cluster and deploy the containers to it, e.g. using a script (bash script on Linux/Mac, or batch file on Windows) that includes all the relevant commands for this procedure.

**Hint:** To avoid too much complexity, only deploy the web app part using kubernetes, and run API service on a single VM. Use the external IP of API in the app's code.