**PROJECT:** TASK SCHEDULER

Used Technologies: Python, mysql, shell scripting

For storage and deployment: Docker, Kubernetes, Jenkins

**Python:** Created a Python script named app.py and implemented the Task Scheduler application using Flask. Python is used in Task Scheduler such as data processing, interacting with the database, and implementing business logic, handle HTTP requests received by the Flask application

**Flask:** Flask is a lightweight Python web framework that provides useful tools and features for creating web applications in the Python Language.

Flask is used in the Task Scheduler project to create a web-based interface for managing tasks. Here are the steps how Flask is typically used in the Task Scheduler project:

1. Import Flask and other required modules

2. Create an instance of the Flask application

3. Define routes and route handlers

4. Render HTML templates

5. Access request data

6. Run the Flask application

**Mysql:** Mysql is used in Task Scheduler to store information.

We implement this by installing ‘mysql-connector-python’

In the Task Scheduler project, MySQL is used as the database to store task-related information. Here's how MySQL is typically used in the project:

1. Establishing a connection

2. Executing queries:

3. Error handling:

4. Integration with Flask:

**Shell Scripting:** ​​Shell scripting is used for automating tasks, executing commands, and managing deployments.

**Docker:** Docker is used to containerize the application and simplify the deployment process by creating a ‘Dockerfile’ with all the instructions to build a docker image of the project.

1. For Task Scheduler, the Dockerfile would look like:

meghanareddy@Meghanas-MacBook-Air ts % cat Dockerfile

# Use a base image

FROM python:3.9

# Set the working directory

WORKDIR /app

# Copy the requirements file

COPY requirements.txt .

# Install the project dependencies

RUN pip install --no-cache-dir -r requirements.txt

# Copy the application code into the image

COPY . .

# Expose the required port

EXPOSE 5000

# Set the command to run the application

CMD ["python", "app.py"]

1. Building the Docker image: using docker build command, we can build a docker image.

‘-t’ used to specify a name which is task-scheduler, ‘:latest’ latest is the tag for that name. Whereas ‘.’ is current directory.

docker build -t task-scheduler:latest .

1. Running the Docker container:

docker run -p 5000:5000 -d task-scheduler:latest

**Kubernetes:**

1. Dockerize the Task Scheduler application:

2. Create a Kubernetes deployment:

task-scheduler-deployment.yaml

3. Create a Kubernetes service:

task-scheduler-service.yaml

4. Deploy the Task Scheduler application on Kubernetes:

kubectl apply -f task-scheduler-deployment.yaml -f task-scheduler-service.yaml

5. Access the Task Scheduler application:

minikube service task-scheduler-service --url

Minikube is a lightweight Kubernetes implementation that creates a VM on your local machine and deploys a simple cluster containing only one node. Minikube is available for Linux, macOS, and Windows systems.

Jenkins: Jenkins can be used to automate the build, test, and deployment processes of the application.

CLONING a Repository in Jenkins:

Dashboard > New Item

Create a Freestyle Project and Name it as ‘task scheduler’

Now, get to the Configure in left side of jenkins menu

In Source Code Management:

Provide Repository URL:

<https://github.com/devimeghana2003/Task-Scheduler.git>

Credentials:

Add - Jenkins

Enter github username and password

Select the Branches to build:

(empty) : any branch

Repository Browser:

githubweb

URL:

<https://github.com/devimeghana2003/Task-Scheduler>

Now, Save and Build the Project to clone the github Repository.

Now look at the code files:

meghanareddy@Meghanas-MacBook-Air ts % cat app.py

from flask import Flask, render\_template, request

import mysql.connector

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

# Database configuration

db\_config = {

'host': 'localhost',

'user': 'root',

'password': 'Devi@1234',

'database': 'task\_scheduler'

}

# Connect to the MySQL database

def connect\_to\_database():

conn = mysql.connector.connect(\*\*db\_config)

return conn

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

def index():

conn = connect\_to\_database()

cursor = conn.cursor()

# Retrieve tasks from the database

query = 'SELECT \* FROM tasks'

cursor.execute(query)

tasks = cursor.fetchall()

# Close the database connection

cursor.close()

conn.close()

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

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

def create\_task():

task\_name = request.form['name']

task\_description = request.form['description']

conn = connect\_to\_database()

cursor = conn.cursor()

# Insert the task into the database

query = 'INSERT INTO tasks (name, description) VALUES (%s, %s)'

values = (task\_name, task\_description)

cursor.execute(query, values)

conn.commit()

# Close the database connection

cursor.close()

conn.close()

return 'Task created successfully'

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

app.run(debug=True)

meghanareddy@Meghanas-MacBook-Air ts %

meghanareddy@Meghanas-MacBook-Air ts % cat requirements.txt

Flask==2.0.1

mysql-connector-python==8.0.26

meghanareddy@Meghanas-MacBook-Air ts % cat Dockerfile

# Use the official Python base image

FROM python:3.9

# Set the working directory in the container

WORKDIR /app

# Copy the requirements file

COPY requirements.txt .

# Install the Python dependencies

RUN pip install --no-cache-dir -r requirements.txt

# Copy the application files

COPY app.py .

COPY templates templates

# Expose the port your Flask application is running on

EXPOSE 5000

# Run the Flask application

CMD ["python", "app.py"]

meghanareddy@Meghanas-MacBook-Air ts % cat task-schedule-deployment.yaml

cat: task-schedule-deployment.yaml: No such file or directory

meghanareddy@Meghanas-MacBook-Air ts % cat task-scheduler-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: task-scheduler-deployment

spec:

replicas: 1

selector:

matchLabels:

app: task-scheduler

template:

metadata:

labels:

app: task-scheduler

spec:

containers:

- name: task-scheduler

image: task-scheduler:latest

ports:

- containerPort: 5000

meghanareddy@Meghanas-MacBook-Air ts % cat task-scheduler-service.yaml

apiVersion: v1

kind: Service

metadata:

name: task-scheduler-service

spec:

selector:

app: task-scheduler

ports:

- protocol: TCP

port: 80

targetPort: 5000

type: LoadBalancer

meghanareddy@Meghanas-MacBook-Air ts %

meghanareddy@Meghanas-MacBook-Air templates % cat index.html

<!DOCTYPE html>

<html>

<head>

<title>Task Scheduler</title>

</head>

<body>

<h1>Task List</h1>

<table>

<tr>

<th>ID</th>

<th>Name</th>

<th>Description</th>

</tr>

{% for task in tasks %}

<tr>

<td>{{ task[0] }}</td>

<td>{{ task[1] }}</td>

<td>{{ task[2] }}</td>

</tr>

{% endfor %}

</table>

<h2>Create Task</h2>

<form action="/create" method="post">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required><br><br>

<label for="description">Description:</label>

<textarea id="description" name="description"

required></textarea><br><br>

<input type="submit" value="Create">

</form>

</body>

</html>

meghanareddy@Meghanas-MacBook-Air ts % docker build -t task-scheduler .

[+] Building 11.5s (11/11) FINISHED

=> [internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> [internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 482B 0.0s

=> [internal] load metadata for docker.io/library/python:3.9 2.4s

=> [1/6] FROM docker.io/library/python:3.9@sha256:ba10a2af9d6c3bd0d20c46 0.0s

=> [internal] load build context 0.0s

=> => transferring context: 187B 0.0s

=> CACHED [2/6] WORKDIR /app 0.0s

=> [3/6] COPY requirements.txt . 0.0s

=> [4/6] RUN pip install --no-cache-dir -r requirements.txt 8.8s

=> [5/6] COPY app.py . 0.0s

=> [6/6] COPY templates templates 0.0s

=> exporting to image 0.2s

=> => exporting layers 0.2s

=> => writing image sha256:6fb67903a3ff8acea89e19d797e49dfc0b0415b3d34af 0.0s

=> => naming to docker.io/library/task-scheduler 0.0s

meghanareddy@Meghanas-MacBook-Air ts % docker run -p 5000:5000 task-scheduler

\* Serving Flask app 'app' (lazy loading)

\* Environment: production

WARNING: This is a development server. Do not use it in a production deployment.

Use a production WSGI server instead.

\* Debug mode: on

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

\* Running on **http://127.0.0.1:5000**

Press CTRL+C to quit

\* Restarting with stat

\* Debugger is active!

\* Debugger PIN: 951-221-595



