Exp 9: Automate the process of running containerized application developed in exercise 7 using Kubernetes.

AIM : Using Kubernetes automate the process of running containerized application developed in exercise 7.

DESCRIPTION: To automate the process of running the containerized application developed in exercise 7 using Kubernetes,
you can follow these steps:

- Create a Kubernetes cluster: Create a Kubernetes cluster using a cloud provider, such as Google Cloud or Amazon Web Services, or using a local installation of Minikube.
- Push the Docker image to a registry: Push the Docker image of your application to a container registry, such as Docker Hub or Google Container Registry.
- Create a deployment: Create a deployment in Kubernetes that specifies the number of replicas and the Docker image to use.

Serivce.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: exp7-service
spec:
  selector:
    app: exp7
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5000
  type: NodePort
```

requirements.txt

```
Flask==2.3.2
```

Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY . .
RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 5000
CMD ["python", "app.py"]
```

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: exp7-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: exp7
  template:
    metadata:
      labels:
        app: exp7
    spec:
      containers:
      - name: exp7-app
        image: cjitscse/exp7-app:latest
        ports:
        - containerPort: 5000
```

app.py

```
from flask import Flask, render_template, request, redirect, url_for
          app = Flask(__name__)
# Temporary storage (replace with a database in production)
users = []
@app.route('/')
def home():
    return redirect(url_for('register'))
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
users.append({'username': username, 'email': email})
        return redirect(url_for('success'))

    return render_template('register.html')
@app.route('/success')
def success():
    return render_template('success.html')
if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

EXPERIMENT NO.: 10. Install and Explore Selenium for automated testing

AIM: Install and Explore Selenium for automated testing

DESCRIPTION:
To install and explore Selenium for automated testing, you can follow these steps:

Install Java Development Kit (JDK):

- Selenium is written in Java, so you'll need to install JDK in order to run it. You can download and install JDK from the official Oracle website.
  - Install the Selenium WebDriver:
- You can download the latest version of the Selenium WebDriver from the Selenium website. You'll also need to download the appropriate driver for your web browser of choice (e.g. Chrome Driver for Google Chrome).

Install an Integrated Development Environment (IDE):

- To write and run Selenium tests, you'll need an IDE. Some popular choices include Eclipse, IntelliJ IDEA, and Visual Studio Code.
- Write a simple test:
- Once you have your IDE set up, you can write a simple test using theSelenium WebDriver:

Main.java:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class Main {
public static void main(String[] args)
{
  System.setProperty("webdriver.chrome.driver","C:\\exp10\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.google.com");
        System.out.println(driver.getTitle());
        driver.quit();
}
}
```

- Run the test:
  - ➢ Create a exp10 folder in the C Drive.
  - ➢ Create a libs folder in the exp10.
  - ➢ Download ChromeDriver.exe from https://googlechromelabs.github.io/chrome-for-testing website.
  - ➢ Download selenium for java .zip file from https://www.selenium.dev/downloads website.
  - ➢ Extract downloaded zip file in to libs folder.

Run the test using your IDE or from the command line using the following command:

Command for Compile the Code:

```
javac -cp ".;libs/*" Main.java

java -cp ".;libs/*" Main
```

Exp11: Write a simple program in JavaScript and perform testing using Selenium

AIM: Write a simple program in JavaScript and perform testing using Selenium

Description:

PROGRAM: Testing the login.html with Simple JavaScript program(contain Test Cases)that you can test using Selenium

login.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
</head>
<body>
  <h2>Login Form</h2>
  <form id="loginForm">
    <label>Username:</label>
    <input type="text" id="username"><br><br>
    <label>Password:</label>
    <input type="password" id="password"><br><br>
    <button type="button" id="loginButton">Login</button>
  </form>

  <p id="message"></p>

  <script>
    document.getElementById("loginButton").addEventListener("click", function() {
      const user = document.getElementById("username").value;
      const pass = document.getElementById("password").value;
      if (user === "suresh" && pass === "12345") {
        document.getElementById("message").innerText = "Login successful!";
      } else {
        document.getElementById("message").innerText = "Invalid credentials!";
      }
    });
  </script>
</body>
</html>
```

testLogin.js:

```
const { Builder, By, until } = require('selenium-webdriver');

(async function loginTest() {
  let driver = await new Builder().forBrowser('chrome').build();

  try {
    // Open your page (use http://localhost/... if using XAMPP)
    await driver.get('http://localhost/exp11/login.html');

    // Wait a bit for the page to load
    await driver.sleep(1000);

    // Enter username and password
    await driver.findElement(By.id('username')).sendKeys('suresh');
    await driver.findElement(By.id('password')).sendKeys('12345');

    // Click login button
    await driver.findElement(By.id('loginButton')).click();

    // Wait until message appears
    let message = await driver.wait(until.elementLocated(By.id('message')), 2000);
    let text = await message.getText();

    console.log("Message displayed:", text);
    if (text === "Login successful!") {
      console.log("✅Test Passed!");
    } else {
      console.log("❌Test Failed!");
    }

  } catch (err) {
    console.error("Test Error:", err);
  } finally {
    await driver.quit();
  }
})(); const { Builder, By, until } = require('selenium-webdriver');

(async function loginTest() {
  let driver = await new Builder().forBrowser('chrome').build();

  try {
    // Open your page (use http://localhost/... if using XAMPP)
    await driver.get('http://localhost/exp11/login.html');

    // Wait a bit for the page to load
    await driver.sleep(1000);

    // Enter username and password
    await driver.findElement(By.id('username')).sendKeys('suresh');
    await driver.findElement(By.id('password')).sendKeys('12345');

    // Click login button
    await driver.findElement(By.id('loginButton')).click();

    // Wait until message appears
    let message = await driver.wait(until.elementLocated(By.id('message')), 2000);
    let text = await message.getText();

    console.log("Message displayed:", text);
    if (text === "Login successful!") {
      console.log("✅Test Passed!");
    } else {
      console.log("❌Test Failed!");
    }

  } catch (err) {
    console.error("Test Error:", err);
  } finally {
    await driver.quit();
  }
})();
```

Run Process Steps:

1. **Create a exp11 folder in the c:\xampp\htdocs folder.**
2. **Copy above two files in the folder.**
3. **Go to command prompt in that folder.**
4. **Type command --- > npm init –y**
5. **Next type command→ npm install selenium-webdriver chromedriver**
6. **After install selenium check the testLogin.js**
7. **Again type the command → node testLogin.js**

Exp12: Develop test cases for the above containerized application using selenium.

AIM: Develop test cases using python script and test the application using selenium.

Description:

Step 1: Prerequisites

1. Python
   - ✓ Check if Python is installed: check using command → python –version

   If not, download from https://www.python.org/downloads.

2. Google Chrome(check the latest version is updated or not):
   - ✓ Make sure Chrome browser is installed.

3. Chrome WebDriver
   - ✓ Go to: https://chromedriver.chromium.org/downloads
   - ✓ Download the version matching your Chrome version.
   - ✓ Extract it and note the **path** (for example: `C:\chromedriver\chromedriver.exe`).

4. Install Selenium

   Type the command in Command Prompt (or Terminal):

   - ✓ pip install selenium

Step 2: Folder Setup:

- ✓ Create a project folder: exp12 in the c:\xampp\htdocs folder.
- ✓ Keep test_login_php.py and chromedriver.exe in the exp12 folder

Step 3: Create program file in exp12 folder:.

Login.php:

```
<html>
<head>
   <title>Simple Login</title>
   <style>
      body {
         font-family: Arial;
         background: #f3f3f3;
      }
      .login-box {
         width: 300px;
         margin: 100px auto;
         background: white;
         padding: 20px;
         border-radius: 10px;
         box-shadow: 0 0 10px gray;
```

```
        }
        input[type=text], input[type=password] {
            width: 100%;
            padding: 8px;
            margin: 8px 0;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        input[type=submit] {
            background-color: #4CAF50;
            color: white;
            padding: 10px;
            border: none;
            cursor: pointer;
            width: 100%;
            border-radius: 4px;
        }
        input[type=submit]:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
<div class="login-box">
    <h2>Login Form</h2>
    <form method="POST">
        <label>Username:</label>
        <input type="text" name="username" required>
        <label>Password:</label>
        <input type="password" name="password" required>
        <input type="submit" name="login" value="Login">
    </form>

    <?php
    if (isset($_POST['login'])) {
        $username = $_POST['username'];
        $password = $_POST['password'];

        // Simple hardcoded check
        if ($username == "admin" && $password == "12345") {
            echo "<p style='color:green;'>Login Successful!</p>";
        } else {
            echo "<p style='color:red;'>Invalid Username or Password</p>";
        }
    }
    ?>
</div>
</body>
</html>
```

test_login_php.py:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

# ---------- CONFIGURATION ----------
URL = "http://localhost/exp12/login.php"   # change if hosted elsewhere
USERNAME = "admin"
PASSWORD = "12345"

# ---------- SETUP CHROME ----------
driver = webdriver.Chrome()
driver.maximize_window()
driver.get(URL)
time.sleep(2)

# ---------- TEST CASE 1: Page Title and URL ----------
print("TC01 - Checking page title and URL")
assert "Login" in driver.title, " ❌Page title mismatch"
assert "login" in driver.current_url, " ❌URL does not contain 'login'"
print("✅Page loaded correctly")

# ---------- TEST CASE 2: Successful Login ----------
print("TC02 - Valid login test")
driver.find_element(By.NAME, "username").clear()
driver.find_element(By.NAME, "username").send_keys(USERNAME)
driver.find_element(By.NAME, "password").clear()
driver.find_element(By.NAME, "password").send_keys(PASSWORD)
driver.find_element(By.NAME, "login").click()
time.sleep(1)

page_source = driver.page_source
assert "Login Successful" in page_source, " ❌Valid login failed"
print("✅Valid login passed")

# ---------- Go back to login page ----------
driver.get(URL)
time.sleep(1)

# ---------- TEST CASE 3: Invalid Username ----------
print("TC03 - Invalid username test")
driver.find_element(By.NAME, "username").send_keys("wronguser")
driver.find_element(By.NAME, "password").send_keys(PASSWORD)
driver.find_element(By.NAME, "login").click()
time.sleep(1)
assert "Invalid" in driver.page_source, " ❌Invalid username not handled"
print("✅Invalid username handled correctly")

# ---------- Go back to login page ----------
driver.get(URL)
time.sleep(1)
```

```python
# ---------- TEST CASE 4: Invalid Password ----------
print("TC04 - Invalid password test")
driver.find_element(By.NAME, "username").send_keys(USERNAME)
driver.find_element(By.NAME, "password").send_keys("wrongpass")
driver.find_element(By.NAME, "login").click()
time.sleep(1)
assert "Invalid" in driver.page_source, " ❌Invalid password not handled"
print("✅Invalid password handled correctly")

# ---------- Go back to login page ----------
driver.get(URL)
time.sleep(1)

# ---------- TEST CASE 5: Empty Fields ----------
print("TC05 - Empty fields test")
driver.find_element(By.NAME, "username").clear()
driver.find_element(By.NAME, "password").clear()
driver.find_element(By.NAME, "login").click()
time.sleep(1)
# Check for 'required' attribute behavior or no action
print("✅Empty fields test executed (browser shows required warning if HTML5 required attribute is used)")

# ---------- TEST CASE 6: Verify Input Elements Exist ----------
print("TC06 - Checking presence of input fields")
username_field = driver.find_element(By.NAME, "username")
password_field = driver.find_element(By.NAME, "password")
login_button = driver.find_element(By.NAME, "login")

assert username_field.is_displayed(), " ❌Username field missing"
assert password_field.is_displayed(), " ❌Password field missing"
assert login_button.is_displayed(), " ❌Login button missing"
print("✅All input fields found")

# ---------- CLEANUP ----------
time.sleep(2)
driver.quit()
print("\n All test cases executed successfully!")
```

Step 4: Run the code

- ✓ **Goto the command prompt type the command:**
  ➔ **py test_login_php.py**