CYCLE 2:

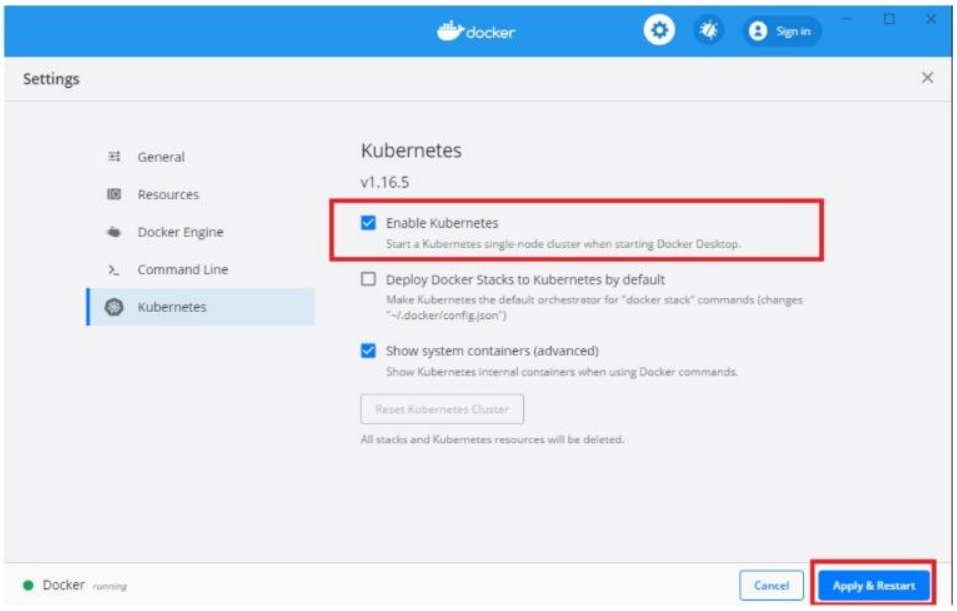
1. Integrate Kubernetes and Docker
2. Automate the process of running containerized applications for exercise 7 using Kubernetes.
3. Install and Explore Karate and Spring boot testing for automated testing.
4. Write a simple program in JavaScript and perform testing using Karate testing.
5. Develop test cases for the above containerized application using Spring boot testing.

cycle-2 exp-1

Kubernetes is an open source orchestration system for automating the management, placement, scaling, and routing of containers, gaining popularity among developers and IT operations teams. Docker Engine, the original container engine for Kubernetes, plays a pivotal role in managing container operations in the host environment.

Install and turn on Kubernetes

1. From the Docker Desktop Dashboard, select the Settings.
2. Select Kubernetes from the left sidebar.
3. Next to Enable Kubernetes, select the checkbox.
4. Select Apply & Restart to save the settings and then select Install to confirm. This instantiates images required to run the Kubernetes server as containers, and installs the /usr/local/bin/kubectl command on your machine.



Kubernetes

v1.16.5

- [✓] Enable Kubernetes
  Start a Kubernetes single-node cluster when starting Docker Desktop.
- [ ] Deploy Docker Stacks to Kubernetes by default
  Make Kubernetes the default orchestrator for "docker stack" commands (changes "~/.docker/config.json")
- [✓] Show system containers (advanced)
  Show Kubernetes internal containers when using Docker commands.

All stacks and Kubernetes resources will be deleted.

Add your own screenshot.

5. Verify your Kubernetes cluster
   You can test the command by listing the available nodes:

cycle-2 exp-2

Step 1: Create a Docker Image

First, create a Docker image for your application. This involves writing a **Dockerfile** that specifies the base image, copies the application code, and sets the command to run the application.

```
FROM python:3.9-slim
# Set the working directory in the container
WORKDIR /app
# Copy the current directory contents into the container at /app
COPY . /app
# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt
# Make port 80 available to the world outside this container
EXPOSE 80
# Define environment variable
ENV NAME World
# Run app.py when the container launches
CMD ["python", "app.py"]
```

Build the Docker image using the Dockerfile:

```
docker build -t my-python-app .
```

Step 2: Push the Docker Image to a Registry

Push the Docker image to a container registry like Docker Hub or Google Container Registry, to pull the image during deployment.

```
docker tag my-python-app:latest <your-docker-hub-username>/my-python-app:latest
docker push <your-docker-hub-username>/my-python-app:latest
```

Step 3: Create a Kubernetes Deployment

Create a YAML file named **deployment.yaml** with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: my-python-app
spec:
 replicas: 3
 selector:
  matchLabels:
   app: my-python-app
 template:
  metadata:
   labels:
    app: my-python-app
  spec:
   containers:
   - name: my-python-app
    image: <your-docker-hub-username>/my-python-app:latest
    ports:
    - containerPort: 80
```

Apply the deployment configuration to your Kubernetes cluster:

```
kubectl apply -f deployment.yaml
```

Step 4: Create a Kubernetes Service

Create a YAML file named **service.yaml** with the following content:

```
apiVersion: v1
kind: Service
metadata:
  name: my-python-app-service
spec:
  selector:
    app: my-python-app
  ports:
  - name: http
    port: 80
    targetPort: 80
  type: LoadBalancer
```

Apply the service configuration to your Kubernetes cluster:

```
kubectl apply -f service.yaml
```

Step 5: Verify the Deployment

Check the status of your deployment and service:

```
kubectl get deployments
kubectl get pods
kubectl get svc
```

Step 6: Access the Application

To access your application, you need to get the external IP address of the service:

```
kubectl get svc my-python-app-service -o
jsonpath='{.status.loadBalancer.ingress.hostname}'
```

Open a web browser and navigate to the external IP address to access your application.

cycle-2 exp-3

Install Karate

☐ 1. Prerequisites

Before installing Karate, make sure these are set up:

Tool	Required	Command to Verify	Expected Output Example
Java JDK 17+	□ **Yes**	**java -version**	**java version "17.0.11"**
Apache Maven	□ **Yes**	**mvn -version**	**Apache Maven 3.9.6**

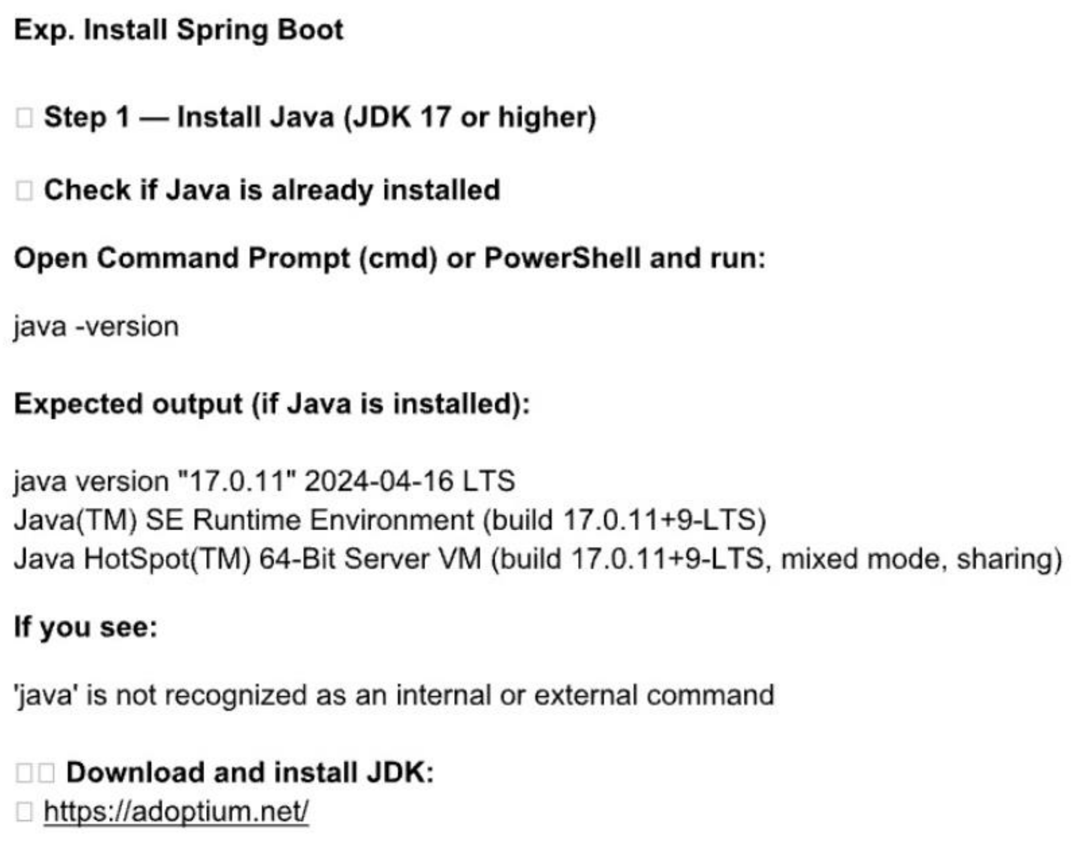
□□ **2. Verify Java**

Open Command Prompt (cmd) or PowerShell, and run:

```
java -version
```

□□ **3. Verify Maven**

```
mvn -version
```



Exp. Install Spring Boot

□ **Step 1 — Install Java (JDK 17 or higher)**

□ **Check if Java is already installed**

Open Command Prompt (cmd) or PowerShell and run:

```
java -version
```

Expected output (if Java is installed):

```
java version "17.0.11" 2024-04-16 LTS
Java(TM) SE Runtime Environment (build 17.0.11+9-LTS)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.11+9-LTS, mixed mode, sharing)
```

If you see:

```
'java' is not recognized as an internal or external command
```

□□ **Download and install JDK:**
□ https://adoptium.net/

During installation:

- Choose "Set JAVA_HOME variable"
- Add to PATH if asked.

Then verify again:

```
java -version
```

□□ Step 2 — Install Apache Maven

□ Check if Maven is installed

```
mvn -version
```

Expected output (if installed):

```
Apache Maven 3.9.6
Maven home: C:\Program Files\apache-maven-3.9.6
Java version: 17.0.11


Default locale: en_US, platform encoding: Cp1252
```

If you see:

```
'mvn' is not recognized as an internal or external command
```

□ Install Maven manually

1. **Go to**: https://maven.apache.org/download.cgi
2. **Download the Binary zip archive** (example: apache-maven-3.9.6-bin.zip)
3. **Extract to:**
   C:\Program Files\Apache\Maven
4. **Add to Environment Variables:**
   - Variable name: MAVEN_HOME
     Value: C:\Program Files\Apache\Maven\apache-maven-3.9.6
   - Add ;%MAVEN_HOME%\bin to your PATH variable.
5. **Restart Command Prompt and verify:**

```
mvn -version
```

cycle-2 exp-4

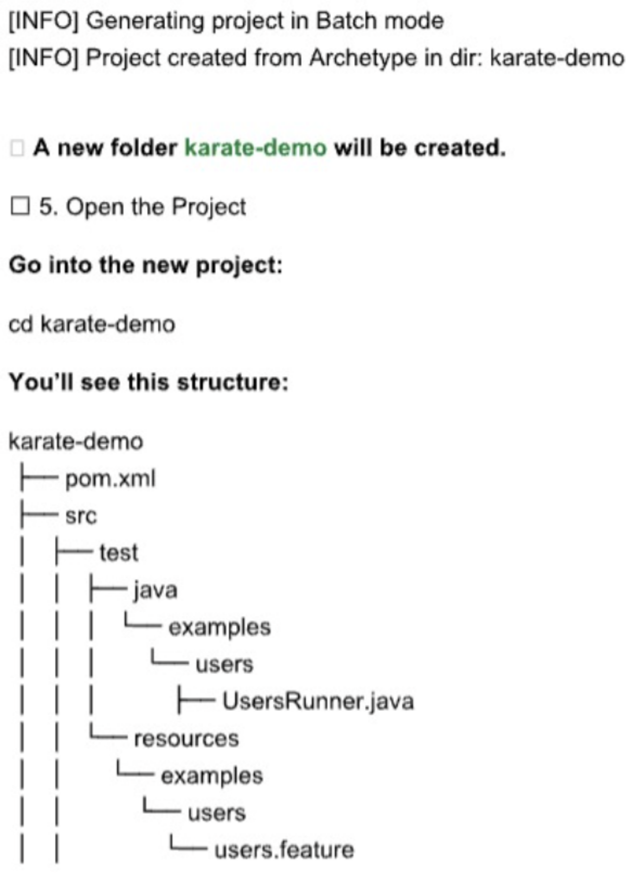
You can create a Karate project quickly using Maven.

Run this in your terminal:

```
mvn archetype:generate \
  -DarchetypeGroupId=com.intuit.karate \
  -DarchetypeArtifactId=karate-archetype \
  -DarchetypeVersion=1.5.0 \
  -DgroupId=com.example \
  -DartifactId=karate-demo \
  -DinteractiveMode=false
```

Expected output:

```
[INFO] Scanning for projects...
```



```
[INFO] Generating project in Batch mode
[INFO] Project created from Archetype in dir: karate-demo
```

▯ **A new folder karate-demo will be created.**

☐ 5. Open the Project

Go into the new project:

```
cd karate-demo
```

You'll see this structure:

```
karate-demo
 ├── pom.xml
 ├── src
 |   ├── test
 |   |   ├── java
 |   |   |   └── examples
 |   |   |       └── users
 |   |   |           ├── UsersRunner.java
 |   |   └── resources
 |   |       └── examples
 |   |           └── users
 |   |               └── users.feature
```

□ **6. Run the Sample Karate Test**

Run this command:

```
mvn test -Dtest=examples.users.UsersRunner
```

Output:

```
---------------------------------------------------
 T E S T S
---------------------------------------------------
Running examples.users.UsersRunner
---------------------------------------------------


 feature: examples/users/users.feature
 scenario: get all users
 * url 'https://jsonplaceholder.typicode.com/users'
 * method get
 * status 200
 -----------------------------------------------------
 BUILD SUCCESS
 -----------------------------------------------------
 Total time: 5.123 s
 Finished at: 2025-10-09T11:15:47+05:30
```

□ **If you see BUILD SUCCESS — Karate is correctly installed and working!**

☐ 7. View Reports

After tests, Karate automatically generates a HTML report.

Check:

```
target/karate-reports/karate-summary.html
```

Open it in your browser — it shows test results in color-coded format.

cycle-2 exp-5

Option 1 — Using Spring Initializr (Recommended)

1. **Open your browser and go to** □ https://start.spring.io
2. **Fill the form:**
   - Project: Maven
   - Language: Java
   - Spring Boot: (latest stable, e.g. 3.3.3)
   - Group: com.example
   - Artifact: demo
   - Dependencies: Spring Web
3. **Click Generate** → it downloads demo.zip.
4. **Extract it to any folder** (for example C:\springboot\demo).

Option 2 — Using Command Line

You can also use:

```
mvn archetype:generate -DgroupId=com.example \

-DartifactId=demo -DarchetypeArtifactId=maven-archetype-quickstart \


-DinteractiveMode=false
```

□□ Step 4 — Run the Spring Boot Application

1. **Open Command Prompt.**
2. **Navigate to your project folder:**

```
cd C:\springboot\demo
```

3. Run the app:

```
mvn spring-boot:run
```

Expected browser output:

```
Hello, Spring Boot!
```