National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Arpit Shah
Student ID: 22208224

School of Computing
National College of Ireland

Supervisor: Prof. Sean Heeney

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Arpit Shah
Student ID:	22208224
Programme:	Cloud Computing **Year:** 2023-2024
Module:	MSC Research Project
Lecturer:	Prof. Sean Heeney
Submission Due Date:	12/08/2024
Project Title:	Empirical Study of Cloud Deployment Strategies: Guiding the choice between Containerization, Traditional and Hybrid Deployment
Word Count:	**2499 Page Count: 14**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Arpit Shah

Date: 12/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Arpit Shah
Student ID: 22208224

1 AWS EC2 Setup

1. Log in to AWS Management Console: Open the AWS Management Console at https://aws.amazon.com/ and sign in with your AWS credentials.
2. Navigate to EC2 Dashboard: From the console dashboard, click on "Services" and then select "EC2" under the "Compute" section.
3. Choose an Amazon Machine Image (AMI): In the "Choose an Amazon Machine Image (AMI)" section, search for "Ubuntu 22.04" and select the appropriate AMI.
4. Choose an Instance Type: Select t3.xlarge as the instance type. This type provides a balance of compute, memory, and network resources.
5. Select a Key Pair
    a. Select an existing key pair or create a new one to securely connect to your instance.
    b. Download the key pair file (.pem file) and keep it safe.
    c. Confirm you have access to the selected key pair, as you will need it to connect to your instance.
6. Configure Instance
    a. Click "Next: Configure Instance Details".
    b. Configure the instance as needed. For basic setup, the default settings are usually sufficient.
7. Add Storage
    a. Click "Next: Add Storage".
    b. Specify the storage size and type. Choose 30 GB as storage.
8. Add Tags
    a. Click "Next: Add Tags".
    b. Add key-value pairs to tag your instance. Tags help manage and identify your resources.
9. Configure Security Group
    a. Click "Next: Configure Security Group".
    b. Create a new security group or select an existing one.
    c. Add rules to allow SSH (port 22) access from your IP address.
10. Review and Launch
    a. Click "Review and Launch".
    b. Review your instance configuration and click "Launch".

2 Connecting to Your EC2 Instance using WSL Ubuntu

1. Open WSL Ubuntu
2. Navigate to the Directory Containing Your **.pem** File:

a. Use the **cd** command to navigate to the directory where your **.pem** file is located

```
cd /mnt/c/path/to/your-key-pair.pem
```

3. Set Permissions for Your Key Pair File
   a. Set the correct permissions for your key pair file to ensure it is not publicly viewable.

```
chmod 400 your-key-pair.pem
```

4. Connect to Your EC2 Instance
   a. Use the **ssh** command to connect to your EC2 instance. Replace **your-key-pair.pem** with the name of your key pair file and **your-ec2-public-ip** with the public IP address of your EC2 instance.

```
ssh -i "your-key-pair.pem" ubuntu@your-ec2-public-ip
```

3 Prometheus Setup

1. Downloading and Installing Prometheus

```
wget
https://github.com/prometheus/prometheus/releases/download/v2.26.0/prometheus-2.26.0.linux-amd64.tar.gz
tar xvf prometheus-2.26.0.linux-amd64.tar.gz
cd prometheus-2.26.0.linux-amd64/
```

2. Configuring Prometheus

```
sudo nano prometheus.yml
```

3. Add the Following Configuration
   a. The configuration should be as shown in figure 1 and if it does not exist, make sure to edit the file.

```
# my global config
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['localhost:9090']
```

4. Save and Exit the editor (Ctrl+X, then Y, then Enter).
5. Running Prometheus
    a. Start Prometheus using the following command

```
/prometheus --config.file=prometheus.yml
```

    b. Prometheus should now be running and accessible at: http://<ec2-public-ip>:9090

4 Grafana Setup

1. Installing Grafana

```
sudo apt-get install -y software-properties-common

sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable
main"

wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -

sudo apt-get update

sudo apt-get install grafana
```

Run the above commands to install Grafana on your EC2 instance.

2. Starting Grafana

```
sudo systemctl start grafana-server

sudo systemctl enable grafana-server
```

Start and enable the Grafana service to run at boot.

3. Accessing Grafana
    a. Open a web browser and navigate to: http://<ec2-public-ip>:3000
    b. Log in with the default credentials:
        i. **Username:** admin
        ii. **Password:** admin
    c. Change the password when prompted.
4. Adding Prometheus as a Data Source in Grafana
    a. In Grafana, click on the **Gear icon (settings)**.
    b. Navigate to **Data Sources** and click on **Add data source**.
    c. Select **Prometheus** from the list.
    d. Enter the URL: http://<ec2-public-ip>:9090 (replace <ec2-public-ip> with your EC2 instance's public IP address).
    e. Click on **Save & Test** to verify the connection.

5 JMeter Setup

1. Installing Java
    a. Update Package Index

```
sudo apt update
```

Run the above command to update the package index.

2. Install OpenJDK

```
sudo apt install openjdk-11-jdk -y
```

Run the above command to install OpenJDK 11 (or the latest version available).

3. Verify Java Installation

```
java -version
```

Run the above command to check the installed Java version.

4. Setting JAVA_HOME Environment Variable
   a. Find Java Installation Path

```
sudo update-alternatives --config java
```

Run the above command and note the path of the selected Java version (e.g., /usr/lib/jvm/java-11-openjdk-amd64).

   b. Set JAVA_HOME

```
nano ~/.bashrc
```

   b. Add the following lines at the end of the file (replace with your actual Java path):

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
```

Save and close the file (press Ctrl+X, then Y, and Enter).

   c. Apply the Changes

```
source ~/.bashrc
```

Run the above command to reload the .bashrc file and apply the changes.

5. Downloading and Installing JMeter

```
wget https://archive.apache.org/dist/jmeter/binaries/apache-
jmeter-5.6.2.tgz
```

Run the above command to download JMeter.

6. Extract the Downloaded File

```
tar -xvzf apache-jmeter-5.6.2.tgz
```

Run the above command to extract the JMeter files.

7. Navigate to JMeter Directory

```
cd apache-jmeter-5.6.2
```

Run the above command to navigate to the JMeter directory.

8. Creating and running a JMeter Test Plan
    a. Create the JMeter Test Plan

```
sudo nano test-plan.jmx
```

    b. Refer figure 2 shows the test plan syntax

    c. Run JMeter Test Plan in Non-GUI Mode

```
./bin/jmeter -n -t /path/to/test-plan.jmx -l
```

Replace /path/to/test-plan.jmx with the path to your JMX file.
Replace /path/to/results.csv with the desired path for the results file.

6 MongoDB

1. Import the MongoDB Public GPG Key
    a. Import the MongoDB GPG key

```
curl -fsSL https://www.mongodb.org/static/pgp/server-7.0.asc |
sudo gpg --dearmor -o /usr/share/keyrings/mongodb-archive-
keyring.gpg
```

2. Create a MongoDB List File
    a. Create a list file for MongoDB

```
echo "deb [ arch=amd64,arm64 signed-
by=/usr/share/keyrings/mongodb-archive-keyring.gpg ]
https://repo.mongodb.org/apt/ubuntu $(lsb_release -cs)/mongodb-
org/7.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-
org-7.0.list
```

3. Install MongoDB Packages
    a. Update the package list

```
sudo apt-get update
```

    b. Install the MongoDB packages

```
sudo apt-get install -y mongodb-org
```

4. Start MongoDB
    a. Start MongoDB

```
sudo systemctl start mongod
```

    b. Verify that MongoDB has started successfully

```
sudo systemctl status mongod
```

5. Access MongoDB Shell
    a. Start the MongoDB shell

```
mongosh
```

b. Connect to a remote MongoDB instance if necessary

```
mongosh "mongodb://<remote-ip>:27017"
```

7 RabbitMQ

RabbitMQ is a message broker that supports multiple messaging protocols

1. Update the Package Index

```
sudo apt-get update
```

2. Install RabbitMQ

```
sudo apt-get install -y rabbitmq-server
```

3. Start and Enable RabbitMQ

```
sudo systemctl start rabbitmq-server
sudo systemctl enable rabbitmq-server
```

4. Verify RabbitMQ is Running

```
sudo systemctl status rabbitmq-server
```

5. Enable RabbitMQ Management Console

```
sudo rabbitmq-plugins enable rabbitmq_management
```

6. Access RabbitMQ Management Console
    a. Navigate to http://<ec2-public-ip>:15672 and log in with default credentials (guest/guest).

8 Docker

1. Navigate to Your Project Directory
    a. Open Command Prompt and navigate to the directory containing your Dockerfile (where you want to build the Docker image)

```
cd path\to\your\project-directory
```

2. Build Docker Images
    a. Build the Docker image using the docker build command. Replace your-dockerhub-username/image-name:tag with your Docker Hub username, image name, and tag

```
docker build -t your-dockerhub-username/image-name:tag .
```

    b. Repeat the above command for each service if you have multiple Dockerfiles in your project.
3. Login to Docker Hub
    a. Log in to your Docker Hub account using

```
docker login
```

    Enter your Docker Hub username and password when prompted.
4. Push Docker Images
    a. Push your Docker images to Docker Hub using the docker push command

```
docker push your-dockerhub-username/image-name:tag
```

Repeat the above command for each image you built.

9 Kompose

Kompose is a tool that helps users to convert Docker compose files in Kubernetes compatible files

1. Install Kompose on Windows
    a. Download Kompose: Go to the Kompose GitHub Releases page and download the latest Windows release. For example,
        i. Download ***kompose-windows-amd64.exe*** for 64-bit Windows systems.
2. Rename and Move Kompose
    a. Rename the downloaded file to kompose.exe.
    b. Move kompose.exe to a directory included in your system's PATH. Typically, this could be C:\Windows\System32.
3. Verify Kompose Installation
    a. Open Command Prompt and run

```
kompose version
```

You should see the version information if Kompose is installed correctly.

4. Convert Docker Compose to Kubernetes
    a. Navigate to Your Docker Compose Directory
        i. Use Command Prompt to navigate to the directory containing your docker-compose.yml file

```
cd path\to\your\docker-compose-directory
```

    b. Convert Docker Compose to Kubernetes Resources
        i. Run the following command to generate Kubernetes resources from the Docker Compose file

```
cd path\to\your\docker-compose-directory
```

This will create several YAML files (deployment.yaml, service.yaml, etc.) in the directory.

5. Copy YAML Files to Your EC2 Instance
    a. Use scp (secure copy) to transfer your YAML files to your EC2 instance. Replace <path-to-yaml> with the path to your local YAML files and <ec2-user> and <ec2-public-ip> with your EC2 instance details:

```
scp -i /path/to/your-key-pair.pem <path-to-yaml> ec2-user@<ec2-
public-ip>:/home/ec2-user/
```

    b. Repeat for each YAML file if necessary
6. Deploy Kubernetes Resources on EC2
    a. Apply each YAML file to your Kubernetes cluster using kubectl

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
# Repeat for other YAML files
```

    b. Verify that the Kubernetes resources are running correctly

```
kubectl get pods
kubectl get services
```

c. Access Your Application

```
kubectl get svc
```

Access your application using the IP and port shown in the output

10 MicroK8s

1. Install RabbitMQ Exporter
   a. Install MicroK8s

```
sudo snap install microk8s --classic
```

   b. Add your user to the MicroK8s group

```
sudo usermod -a -G microk8s $USER
sudo chown -f -R $USER ~/.kube
newgrp microk8s
```

   c. Check the MicroK8s status

```
microk8s status --wait-ready
```

2. Enable Prometheus and Grafana
   a. Enable Prometheus, Grafana, and DNS

```
microk8s enable prometheus dashboard dns
```

3. Verify the services
   a. Check the running pods in the observability namespace

```
microk8s kubectl get pods -n observability
```

   b. Check the services

```
microk8s kubectl get svc -n observability
```

4. Accessing Grafana
   a. Get the Cluster IP of Grafana service

```
microk8s kubectl get svc -n observability kube-prom-stack-
```

Access Grafana by navigating to the IP and port shown in the command above, typically http://<Cluster-IP>:3000.
Login with default credentials admin/admin.

11 Node Exporter

Node Exporter is used to expose machine metrics in Prometheus format

1. Install Node Exporter
   a. Download the Node Exporter package

```
wget
https://github.com/prometheus/node_exporter/releases/download/v
1.3.1/node_exporter-1.3.1.linux-amd64.tar.gz
```

   b. Extract the downloaded file

```
tar xvf node_exporter-1.3.1.linux-amd64.tar.gz
```

c. Move into the directory

```
cd node_exporter-1.3.1.linux-amd64/
```

2. Run Node Exporter
    a. Start Node Exporter

```
./node_exporter
```

    b. To keep it running in the background, you can use

```
nohup ./node_exporter &
```

Verify that Node Exporter is running by visiting: http://<ec2-public-ip>:9100/metrics

12 MongoDB Exporter

MongoDB Exporter helps in exporting MongoDB metrics in Prometheus format

1. Install MongoDB Exporter
    a. Download MongoDB Exporter

```
wget
https://github.com/percona/mongodb_exporter/releases/download/v
0.20.4/mongodb_exporter-0.20.4.linux-amd64.tar.gz
```

    b. Extract the downloaded file

```
tar xvf mongodb_exporter-0.20.4.linux-amd64.tar.gz
```

    c. Move into the directory

```
cd mongodb_exporter-0.20.4.linux-amd64/
```

2. Run MongoDB Exporter

```
./mongodb_exporter --mongodb.uri=mongodb://<mongodb-
username>:<mongodb-password>@<ec2-public-ip>:27017
```

Verify by visiting: http://<ec2-public-ip>:9216/metrics

13 RabbitMQ Exporter

RabbitMQ Exporter is used to export RabbitMQ metrics in Prometheus format

1. Install RabbitMQ Exporter
    1. Pull the RabbitMQ Exporter Docker image

```
sudo docker pull kbudde/rabbitmq-exporter:latest
```

    2. Run the RabbitMQ Exporter

```
sudo docker run -d --name rabbitmq-exporter \
  -p 9419:9419 \
  -e RABBIT_USER="admin" \
  -e RABBIT_PASSWORD="admin" \
  -e RABBIT_URL="http://<ec2-public-ip>:15672" \
  kbudde/rabbitmq-exporter:latest
```

3. Check if RabbitMQ Exporter is running

```
curl http://<ec2-public-ip>:9419/metrics
```

14 Traditional Deployment Setup

In traditional deployment setup, all components of the application, including databases and RabbitMQ are installed directly on the EC2 instance without using any containers or Kubernetes.

1. Prerequisites
    a. AWS EC2 instance with Ubuntu 22.04
    b. SSH access to the EC2 instance (refer to Section 2)
2. Installing and Configuring Components

This setup requires all softwares and services to be installed directly on the EC2 instance.

    a. **Prometheus:** Refer to Section 3.
    b. **Grafana:** Refer to Section 4.
    c. **JMeter:** Refer to Section 5.
    d. **MongoDB:** Refer to Section 6.
    e. **Node Exporter:** Refer to Section 11.
    f. **MongoDB Exporter:** Refer to Section 12.
    g. **RabbitMQ:** Refer to Section 7.
    h. **RabbitMQ Exporter:** Refer to Section 13**.**
3. Running the Application
    a. After installing and configuring all necessary components, start each service manually using the commands provided in their respective sections.
    b. **Application Deployment:** For details about the applications being deployed, including access to their source code, refer to Section 18**.**

15 Containerized Deployment Setup

In the containerized deployment setup, the application with its dependencies is containerized and managed using microK8s which is a lightweight Kubernetes distribution.

1. Prerequisites
    a. MicroK8s installed on your EC2 instance (refer to Section 10).
    b. Docker installed on the EC2 instance (refer to Section 8).
2. Building and Pushing Docker Images
    a. Build Docker Images
        i. Refer to Section 8 for steps to build Docker images for each service.
    b. Push Docker Images to Docker Hub
        i. Push the Docker images to Docker Hub (refer to Section 8).
3. Enable Necessary Add-ons in MicroK8s
    a. Enable Prometheus, Grafana, and DNS as detailed in Section 10.
4. Deploy the Application on MicroK8s
    a. Apply the Kubernetes resources using MicroK8s

```
microk8s kubectl apply -f deployment.yaml
microk8s kubectl apply -f service.yaml
# Apply other YAML files as needed
```

b. Verify that all resources are running correctly

```
microk8s kubectl get pods
microk8s kubectl get services
```

5. Accessing the Application
    a. Access the application through the services exposed by MicroK8s. Use kubectl get svc to retrieve service IPs and ports.
    b. **Application Deployment:** For details about the applications being deployed, including access to their source code, refer to Section 18**.**

16 Hybrid Deployment Setup

In Hybrid Deployment Setup, MicroK8s is used to manage containerization and other services like MongoDB and RabbitMQ are installed directly on the EC2 instance.

1. Prerequisites
    a. MicroK8s installed on the EC2 instance (refer to Section 10).
    b. Docker installed on the EC2 instance (refer to Section 8).
2. Installing and Configuring Direct Services
    a. **MongoDB:** Refer to Section 6.
    b. **RabbitMQ:** Refer to Section 7.
3. Deploying Containerized Components with MicroK8s
    a. Install and Configure MicroK8s
        i. Follow the setup and configuration steps in Section 10**.**
        ii. For more information on MicroK8s, visit the official MicroK8s documentation (Canonical Ltd., 2024).
    b. Enable Add-ons in MicroK8s
        i. Enable Prometheus, Grafana, and DNS as detailed in Section 10.
    c. Deploy Kubernetes Resources
        i. Convert Docker Compose files to Kubernetes YAML files using Kompose (refer to Section 9).
        ii. Transfer the YAML files to your EC2 instance.
        iii. Deploy the resources using MicroK8s

```
microk8s kubectl apply -f deployment.yaml
microk8s kubectl apply -f service.yaml
# Apply other YAML files as needed
```

4. Integrating the Environment
    a. Configure Connections
        i. Ensure the containerized application components can communicate with MongoDB and RabbitMQ using the EC2 private IP address.
    b. Monitor and Manage
        i. Use Prometheus and Grafana (enabled in MicroK8s) to monitor the entire environment.
5. Verifying the Setup
    a. Check All Services
        i. Use microk8s kubectl get pods and systemctl status to ensure all services are running.
    b. Access the Application

i. Use the IP addresses and ports exposed by MicroK8s to access the application.
ii. **Application Deployment:** For details about the applications being deployed, including access to their source code, refer to Section 18**.**

17 Deployment Strategy Recommendation Tool

1. Prerequisites
    a. Python 3.x installed on your local machine or EC2 instance.
    b. Necessary Python packages installed (pandas, scikit-learn, joblib, streamlit).
2. Install Required Packages
    a. Open a terminal and run the following command to install the necessary Python packages

```
pip install pandas scikit-learn joblib streamlit
```

3. Create the Dataset
    a. Ensure you have a CSV file named deployment_recommendations.csv with the following columns: **Application, Performance, Scalability, Cost, Reliability, Operational Complexity, and Recommended Deployment**.
4. Populate the Dataset
    a. Fill in the dataset with the characteristics of different applications and their corresponding recommended deployment strategies.
5. Run the Training Script
    a. Use the provided script to train the model and save it as model.pkl

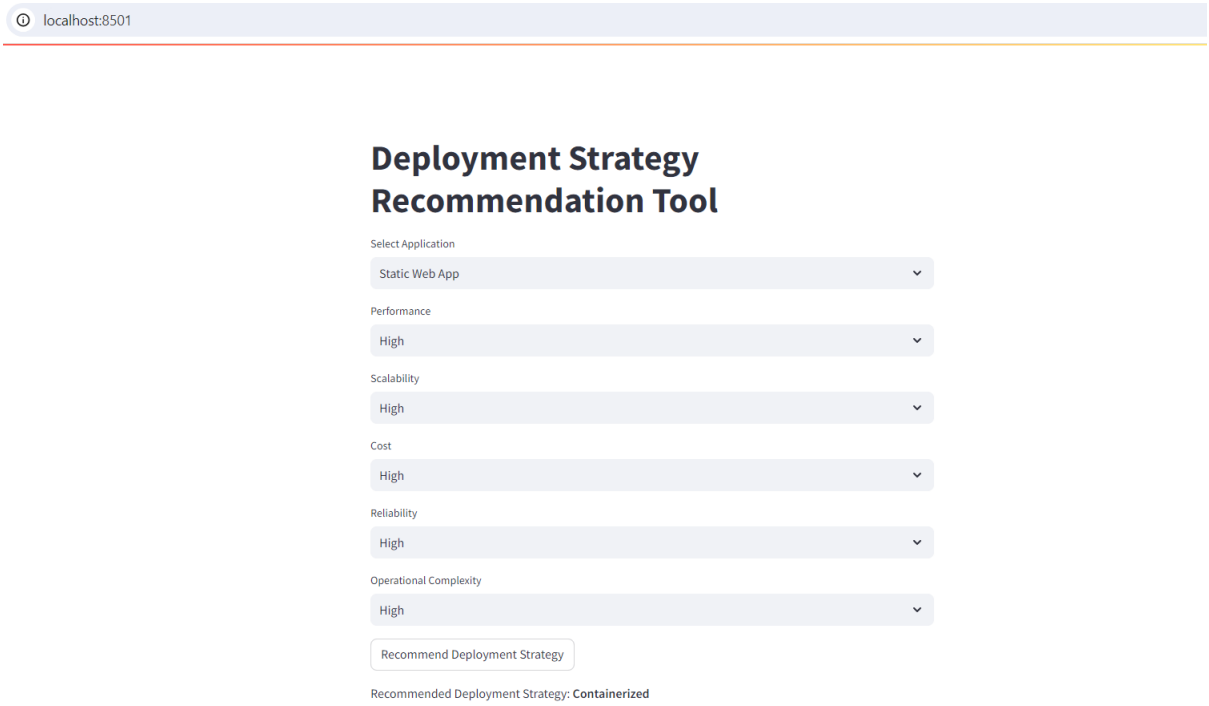
```
python train_model.py
```

    b. Verify the Model
        i. Ensure that model.pkl is generated successfully and contains the trained model.
6. Start the Streamlit Application
    a. In your terminal, run the following command

```
streamlit run app.py
```

    b. Access the Application
        i. After running the command, Streamlit will provide a local URL (usually http://localhost:8501/). Open this URL in your web browser to access the tool.
7. Using the Tool
    a. Select Application Characteristics
        i. Use the dropdown menus to select the characteristics of your application (e.g., Application type, Performance, Scalability).
    b. Get Deployment Recommendation
        i. Click the "Recommend Deployment Strategy" button to receive a recommended deployment strategy based on the input characteristics.

8. User Interface



9. Accessing the Source Code
    a. The source code for the Deployment Strategy Recommendation Tool can be found in the following GitHub repository (Shah, 2024c).

18 Applications Deployed Using Different Strategies

This section provides an overview of the applications that are deployed using the Traditional, Containerized, and Hybrid deployment strategies outlined in this manual. Each application has its source code hosted on GitHub, where you can access, review, and download the code for deployment.

1. Application 1: Static Web App
    a. **Deployment Strategies:** Traditional, Containerized
    b. **Repository:** For the source code, refer to Shah (2024a).
2. Application 2: Database Application
    a. **Deployment Strategies:** Traditional, Containerized, Hybrid
    b. **Description:** For the source code, refer to Knaopel (2023).
3. Application 3: Multithreaded Application with RabbitMQ
    a. **Deployment Strategies:** Traditional, Containerized, Hybrid
    b. **Repository:** For the source code, refer to Shah (2024b).
4. Accessing the Source Code
    a. The source code for each application is available on GitHub. The links provided above will direct you to the respective repositories, where you can clone or download the code.

b. Detailed instructions for deploying these applications using the Traditional, Containerized, and Hybrid strategies are provided in the relevant sections of this manual.

References

AWS Documentation, 2024. *Amazon EC2 Documentation*. Available at: https://docs.aws.amazon.com (Accessed: 11 August 2024).

Apache JMeter, 2024. *JMeter Documentation*. Available at: https://jmeter.apache.org/usermanual/get-started.html (Accessed: 11 August 2024).

Canonical Ltd., 2024. *MicroK8s documentation*. Available at: https://microk8s.io/docs (Accessed: 11 August 2024).

Docker Documentation, 2024. *Docker Documentation*. Available at: https://docs.docker.com (Accessed: 11 August 2024).

Grafana, 2024. *Grafana Documentation*. Available at: https://grafana.com/docs/ (Accessed: 11 August 2024).

Knaopel, 2023. *Dockerized Frontend, Backend, and Database*. GitHub repository. Available at: https://github.com/knaopel/docker-frontend-backend-db (Accessed: 11 August 2024).

Kubernetes Kompose, 2024. *Kompose Documentation*. Available at: https://kompose.io/ (Accessed: 11 August 2024).

MongoDB Documentation, 2024. *MongoDB Documentation*. Available at: https://www.mongodb.com/docs/manual/ (Accessed: 11 August 2024).

Prometheus, 2024. *Prometheus Documentation*. Available at: https://prometheus.io/docs (Accessed: 11 August 2024).

RabbitMQ, 2024. *RabbitMQ Documentation*. Available at: https://www.rabbitmq.com/documentation.html (Accessed: 11 August 2024).

Shah, A**.,** 2024a. *Static Web Application*. GitHub repository. Available at: https://github.com/arpitpshah/static-web-application (Accessed: 11 August 2024).

Shah, A., 2024b. *Multithreaded Application*. GitHub repository. Available at: https://github.com/arpitpshah/multithreaded-application (Accessed: 11 August 2024).

Shah, A., 2024c. *Deployment Strategy Recommendation Tool*. GitHub repository. Available at: https://github.com/arpitpshah/recommendation-tool (Accessed: 11 August 2024).