



Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

Faculty of Cyber Physical System

Dept. of Internet of Things and Robotics Engineering (IRE)

Course Title: Cloud Computing

Course Code: ICT 4359

## **Final Project**

**Submitted to-**

**Teacher name:** Mahir Mahbub

**Submitted by-**

Md. Shahriar Hossain Apu (1901036)

Mehdi Hasan Emon (1901031)

Mehjabin Ashka (1901048)

Arifuzzaman Ripon (1901038)

Date of Submission: 08-11-2023

## **Abstract**

This lab report explores the practical implementation of cloud computing using Docker containers and LocalStack. Cloud computing is a key technology in modern IT infrastructures, and LocalStack provides a convenient and cost-effective way to emulate cloud services locally. This report details the setup, benefits, and results of this cloud computing implementation.

## Contents

<b>Introduction</b>	4
<b>Objectives</b>	4
<b>System Design</b>	4
<b>Materials and Methods</b>	5
Materials	5
Methods	5
<b>Results</b>	8
<b>Benefits</b>	8
<b>Limitations</b>	9
<b>Discussion</b>	9
<b>Conclusion</b>	9
<b>References</b>	10

# Project Name: Implementing Cloud Computing Infrastructure with Docker and LocalStack

## Introduction

Cloud computing has transformed the IT landscape, offering scalable and flexible services that can be accessed remotely. However, testing and development in the cloud can incur costs and may not be ideal for all scenarios. LocalStack is a tool that allows developers to emulate AWS services on their local environments using Docker containers.

## Objectives

- Setup LocalStack using Docker.
- Deploy AWS services locally.
- Test cloud computing concepts using LocalStack.
- Evaluate the benefits and limitations of LocalStack for development and testing.

## System Design

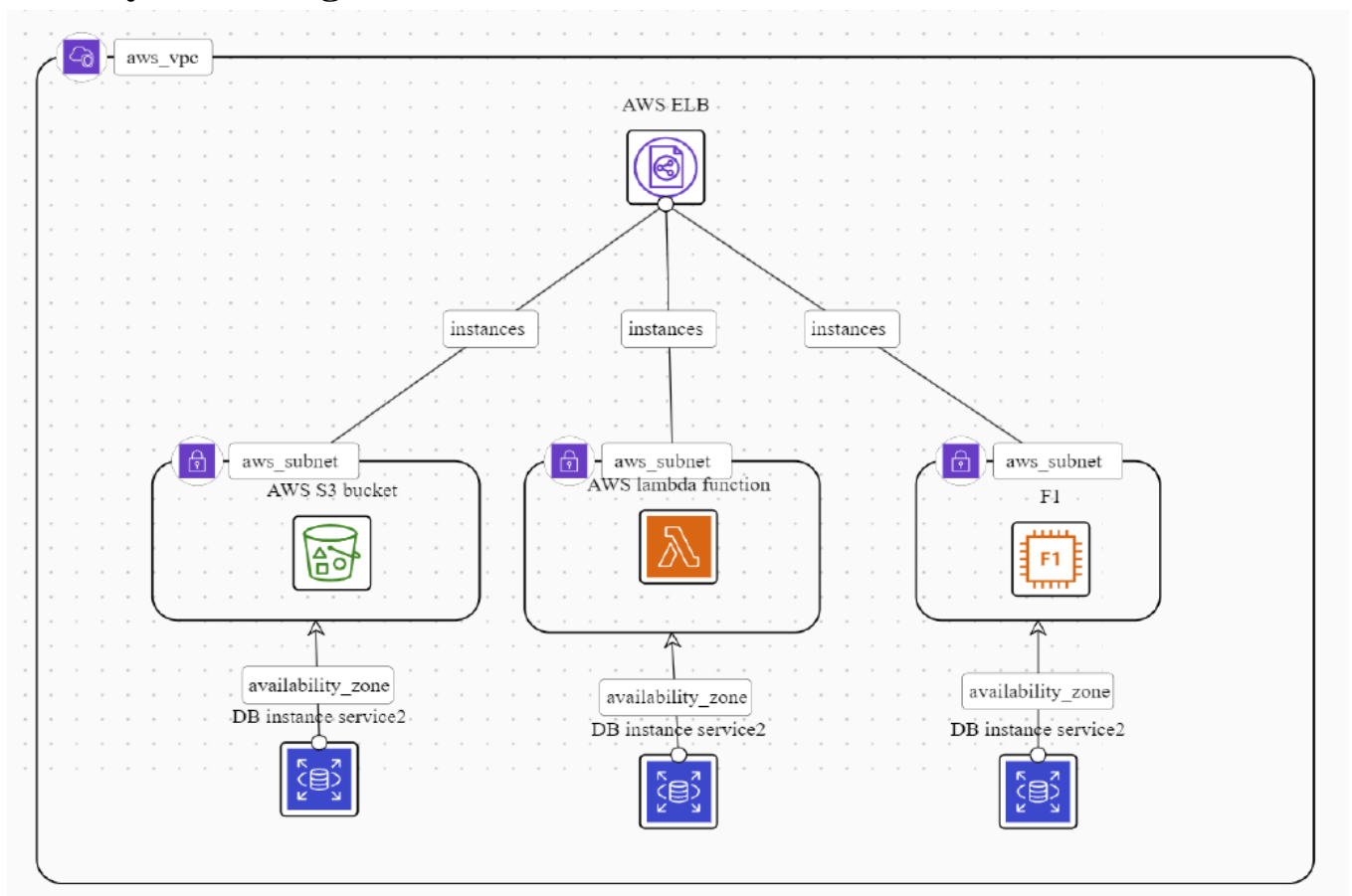


Figure: System Design

## Materials and Methods

### Materials

- Docker
- LocalStack
- AWS CLI
- Terraform (optional)
- AWS SDKs (optional)

### Methods

1. **Docker Setup:** Install Docker on a local development machine if not already installed.

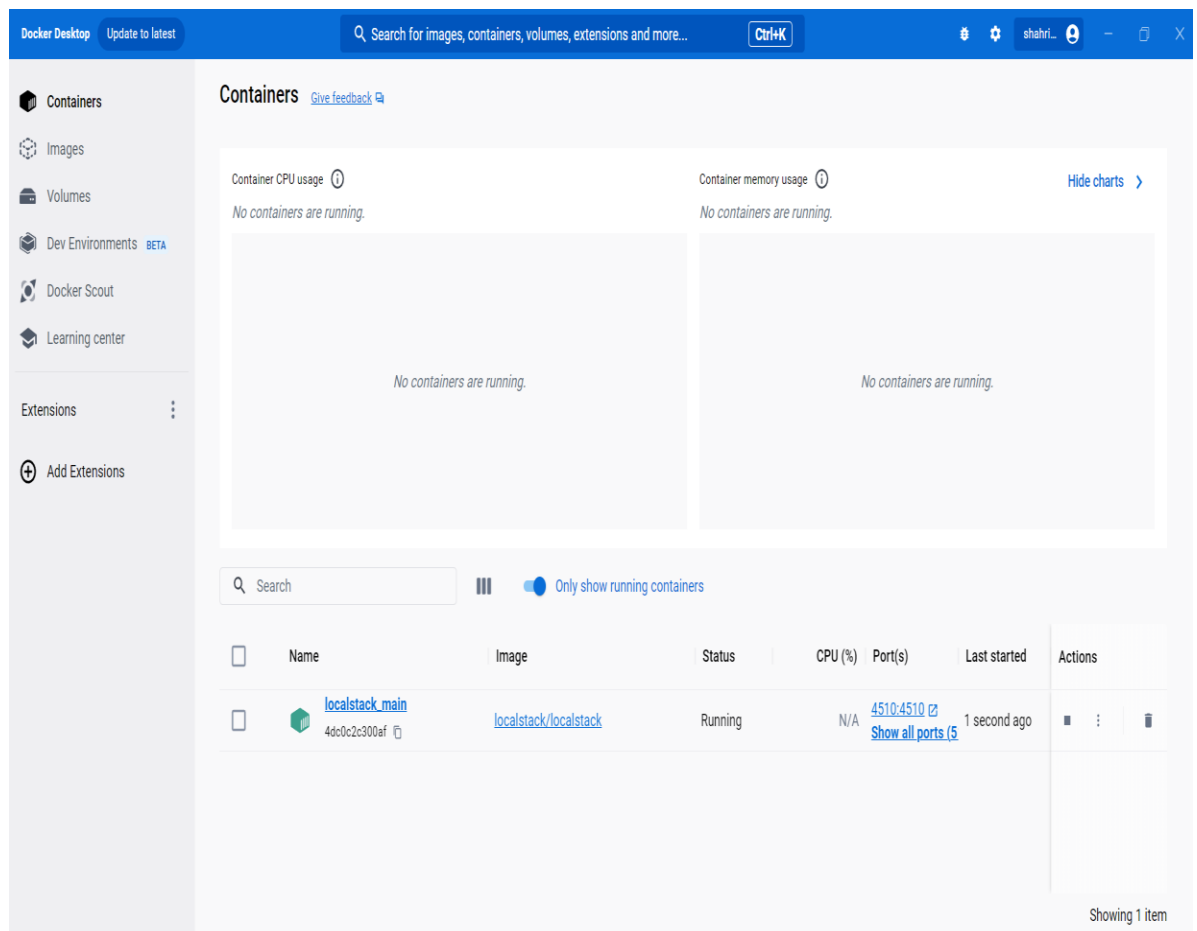
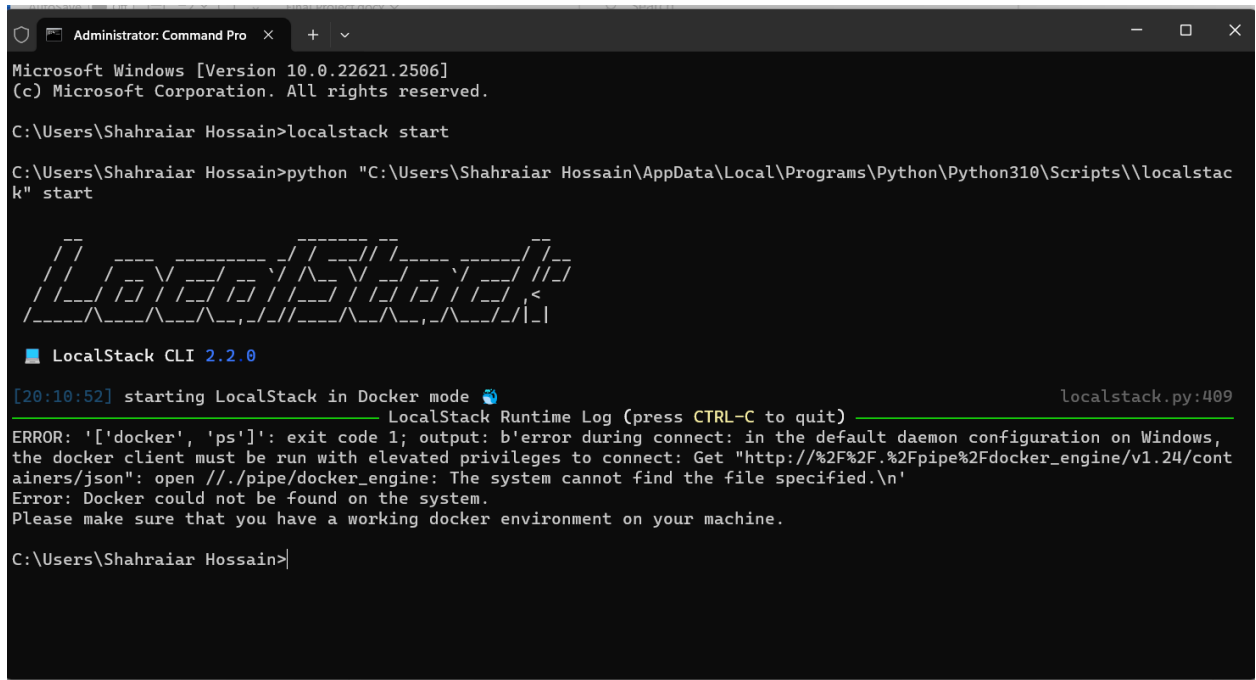


Figure: Docker Dashboard

**2. LocalStack Installation:** Pull and run the LocalStack Docker image, exposing the required ports and configuring the services needed for your use case.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22621.2506]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shahraiar Hossain>localstack start

C:\Users\Shahraiar Hossain>python "C:\Users\Shahraiar Hossain\AppData\Local\Programs\Python\Python310\Scripts\localstack" start

LocalStack CLI 2.2.0

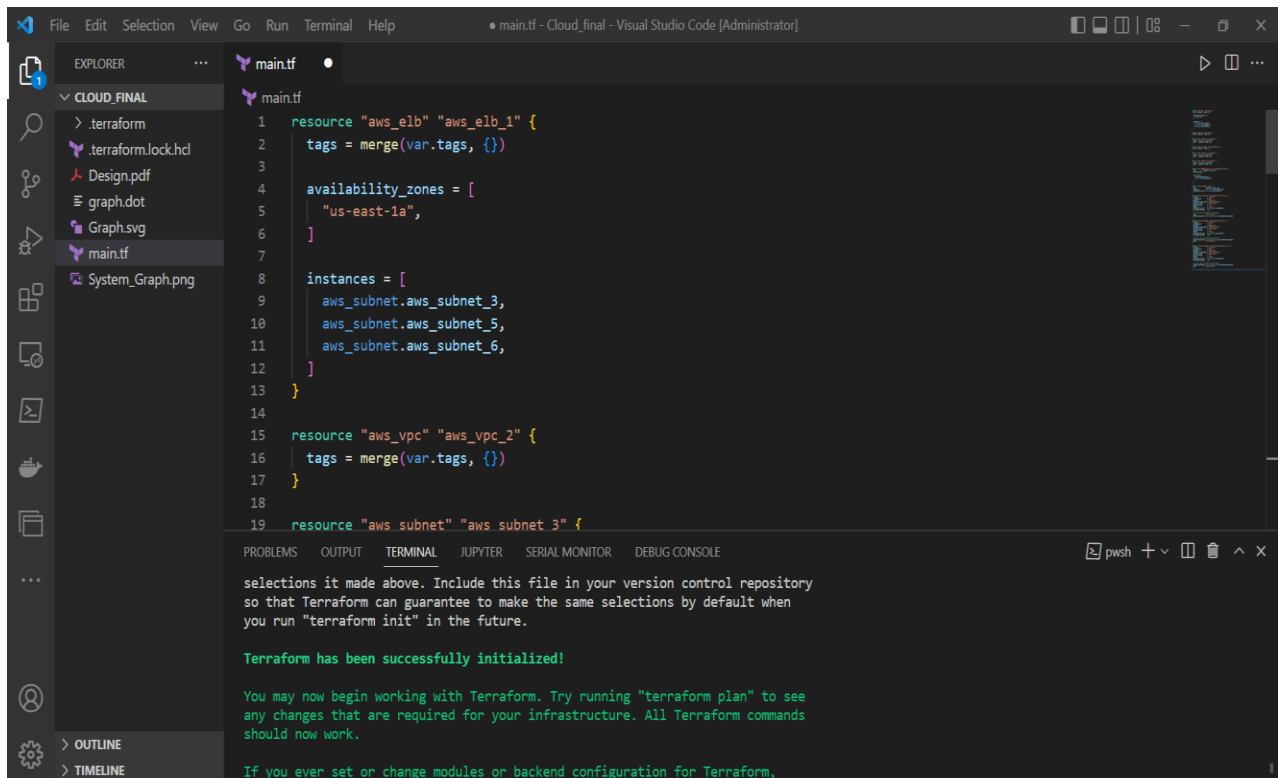
[20:10:52] starting LocalStack in Docker mode
LocalStack Runtime Log (press CTRL-C to quit)
ERROR: '['docker', 'ps']': exit code 1; output: b'error during connect: in the default daemon configuration on Windows, the docker client must be run with elevated privileges to connect: Get "http://pipe/docker_engine/v1.24/containers/json": open //pipe/docker_engine: The system cannot find the file specified.\n'
Error: Docker could not be found on the system.
Please make sure that you have a working docker environment on your machine.

C:\Users\Shahraiar Hossain>
```

Figure: Local Stack Initialization

**3. AWS CLI Configuration:** Configure the AWS CLI to use LocalStack as the AWS endpoint.

```
aws configure set aws_access_key_id test
aws configure set aws_secret_access_key test
aws configure set default.region us-east-1
aws configure set endpoint_url http://localhost:4566
```



The screenshot shows the Visual Studio Code interface with a file explorer on the left containing a project named 'CLOUD\_FINAL'. The main editor displays the 'main.tf' file with Terraform configuration for an AWS environment. The terminal at the bottom shows the output of the 'terraform init' command, indicating successful initialization.

```

1 resource "aws_elb" "aws_elb_1" {
2   tags = merge(var.tags, {})
3
4   availability_zones = [
5     "us-east-1a",
6   ]
7
8   instances = [
9     aws_subnet.aws_subnet_3,
10    aws_subnet.aws_subnet_5,
11    aws_subnet.aws_subnet_6,
12  ]
13 }
14
15 resource "aws_vpc" "aws_vpc_2" {
16   tags = merge(var.tags, {})
17 }
18
19 resource "aws_subnet" "aws_subnet_3" {

```

PROBLEMS OUTPUT TERMINAL JUPYTER SERIAL MONITOR DEBUG CONSOLE

selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future.

**Terraform has been successfully initialized!**

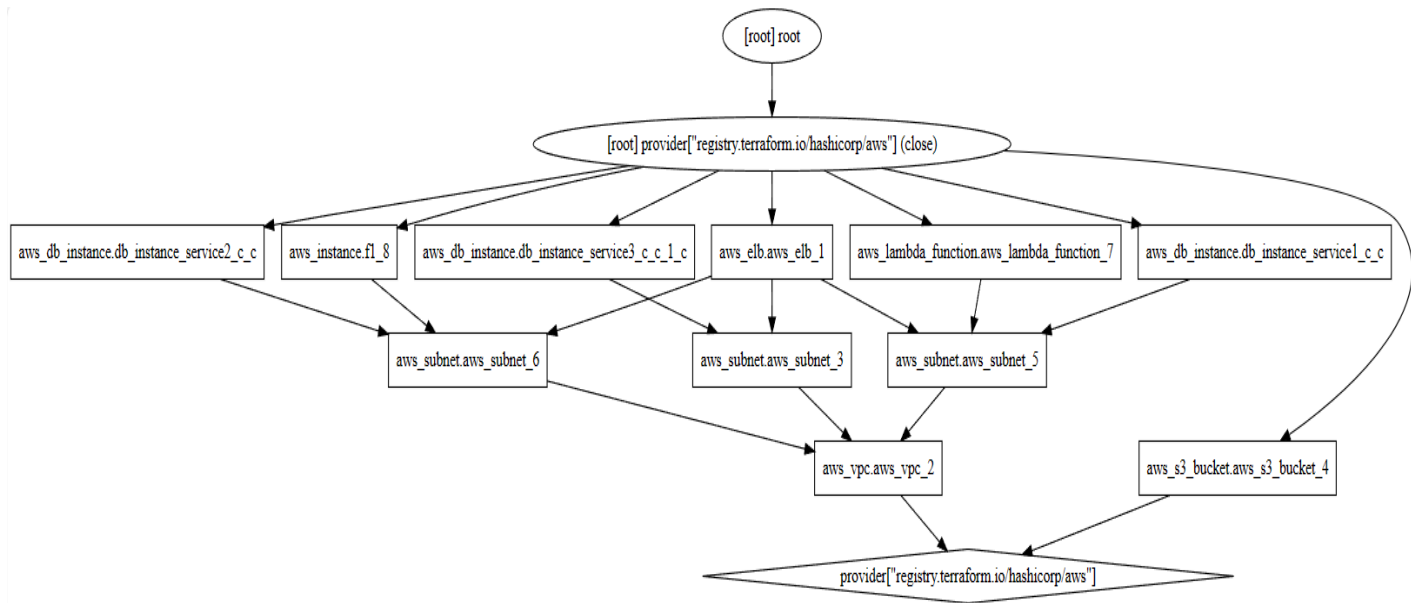
You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform,

Figure: Terraform Coding

**4. Deployment and Testing:** Deploy AWS resources using Terraform or other provisioning tools and test your cloud computing applications against LocalStack.

## Results



**Figure: System Generated Graph**

## Benefits

### 1. Cost Efficiency:

- Pay-as-you-go model reduces capital expenditure.
- Eliminates the need for upfront hardware and infrastructure investments.
- Reduces operational and maintenance costs.

### 2. Scalability:

- Easily scale resources up or down to meet demand.
- Accommodates rapid business growth or seasonal fluctuations.

### 3. Flexibility and Agility:

- Provides flexibility to choose and change services as needed.
- Accelerates deployment of new applications and features.

### 4. Accessibility:

- Access data and applications from anywhere with an internet connection.
- Supports remote work and collaboration.

### 5. Reliability and Availability:

- High uptime and availability through redundancy and failover.
- Cloud providers offer Service Level Agreements (SLAs) for reliability.

### 6. Security:

- Cloud providers invest in robust security measures and compliance.
- Enables data encryption, access controls, and identity management.



### **7. Disaster Recovery and Backup:**

- Automatic backups and recovery mechanisms in place.
- Reduces data loss and downtime in case of disasters.

### **8. Global Reach:**

- Cloud providers have data centers worldwide.
- Expands the reach of applications and services.

### **9. Environmental Impact:**

- Reduces the carbon footprint by optimizing resource utilization.
- Supports sustainability and green computing.

### **10. Automatic Updates and Maintenance:**

- Cloud providers handle software updates and maintenance.
- Ensures the latest security patches and features.

## **Limitations**

**1. Service Coverage:** LocalStack may not support all AWS services, so complex applications may require interactions with the actual AWS cloud.

**2. Real-World Latency:** LocalStack may not fully replicate real-world latencies experienced in the cloud.

## **Discussion**

Implementing cloud computing with LocalStack and Docker is a practical solution for cost-effective and efficient cloud development and testing. It offers a controlled environment, allowing developers to experiment with AWS services without the associated costs. However, users should be aware of its limitations, including service coverage and differences in latency.

Developers can use LocalStack as a part of their CI/CD pipelines, enabling automated testing and deployment of cloud applications. It also allows for offline development and testing in various network conditions, enhancing resilience and robustness of applications.

## **Conclusion**

The implementation of cloud computing using Docker and LocalStack is a valuable tool for developers and teams looking to reduce AWS-related development costs and improve testing capabilities. While not a full replacement for the AWS cloud, LocalStack offers a convenient way to emulate cloud services locally, supporting efficient and cost-effective cloud development.

## References

1. [LocalStack GitHub Repository](<https://github.com/localstack/localstack>)
2. [Docker Official Website](<https://www.docker.com/>)
- 3.[Aws Official Website](<https://aws.amazon.com/>)