**CHHATTISGARH SWAMI VIVEKANAND TECHNICAL UNIVERSITY**

**(UNIVERSITY TEACHING DEPARTMENT)**

8th SEMESTER

# Cloud Computing Lab Manual

**Guided By**
Mr. Shesh Narayan Sahu
Assistant Professor
CSE (DS)

**Prepared By**
Madhurima Rawat
Roll No: 300012821042
Enrolment No: CB4689

# Table of Contents

# Experiment 1

**Aim:** In this experiment, students will learn to understand the basics of cloud computing by setting up and configuring a virtual machine on a cloud platform such as AWS, Google Cloud, or Microsoft Azure.

## Prerequisites:

- **Docker** – A platform for developing, shipping, and running applications in containers.
- **LocalStack** – A fully functional local cloud service emulator for AWS development.
- **AWS CLI** – A command-line tool for managing AWS services and automation.
- **System Requirements** – A computer with **8GB RAM** and at least **16GB free space**.

## Key Concepts:

### Virtual Machine (VM)

- **Definition:** Software-based emulation of a physical computer.
- **Isolation:** Runs in a secure, independent environment.
- **Use Case:** Ideal for cloud computing, testing, and deployment.

### LocalStack

- **Purpose:** Locally simulates AWS cloud services.
- **Benefits:** Enables offline development and testing.
- **Integration:** Supports automation and CI/CD workflows.

### AWS CLI

- **Function:** Command-line tool for managing AWS services.
- **Automation:** Enables scripting and infrastructure as code.
- **Use Case:** Efficient for deployment, monitoring, and automation.

### Docker

- **Concept:** Platform for containerizing applications.
- **Efficiency:** Ensures consistency across environments.
- **Use Case:** Simplifies development, testing, and deployment.

## Installation Guide:

### Docker Setup

1. **Install Docker**: Download and install Docker Desktop from the official website. Enable WSL 2 if required.

2. **Verify Installation**: Run docker --version to confirm the installation.
3. **Start Docker**: Launch Docker Desktop and ensure it is running.
4. **Enable WSL 2 (If Needed)**: Enable it from Docker Desktop settings under "General".

**LocalStack Installation:**

1. **Install LocalStack**:
   - Via pip: pip install localstack
   - Via Docker: docker pull localstack/localstack
2. **Start LocalStack**:
   - Using Docker: docker run -d -p 4566:4566 localstack/localstack
   - If installed locally: localstack start
3. **Install AWS CLI**: pip install awscli
4. **Configure AWS CLI**:
   - Run aws configure with test credentials.
   - Setx AWS_ENDPOINT_URL=http://localhost:4566 for LocalStack.

**AWS CLI Output Formats**

- **JSON** (default): Structured format, ideal for automation.
- **Table**: Human-readable, good for quick reviews.
- **Text**: Simple, suitable for scripts.
- Change format using aws configure or --output flag.

**Troubleshooting**

- **LocalStack Connection Issues**: Run docker ps to check running containers. Restart LocalStack if needed.
- **Long Path Issue on Windows**:
  1. Enable long paths via Registry (HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem).
  2. Enable via Group Policy (for Pro/Education editions).
  3. Restart the system and retry pip install localstack.

## Starting Services:

**1. Ensure the Service is Running**
If LocalStack or another AWS emulator is being used, it's essential to confirm that it has started successfully.

For LocalStack:
localstack start

For Docker (if LocalStack is running via Docker):
docker run -p 4566:4566 -d localstack/localstack

After executing this, the following command can be used to confirm the container is running:
docker ps

**2. Verify Port Availability**
To check if another service is using port 4566, the following command should be run:
netstat -ano | find "4566"

If no output is returned, it indicates the port is free.
If output is returned, it's necessary to identify which process is using the port by executing:
tasklist /FI "PID eq <PID>"
Replace <PID> with the process ID found from the netstat output.

**3. Check Firewall or Antivirus Settings**
It should be ensured that no firewall or antivirus software is blocking connections to port 4566. Temporarily disabling the firewall or antivirus can help in testing.

**4. Confirm localhost Works**
To check if localhost is working correctly on the system, the following command can be used:
ping localhost

**LocalStack Container Image Download Process**
The following output indicates that LocalStack is attempting to pull the container image localstack/localstack to run in Docker mode, which suggests that the setup is working up to this point. Here's what's happening:

**C:\Users\rawat>docker images**
**REPOSITORY TAG IMAGE ID CREATED SIZE**
**Pulling container image localstack/localstack**

**What’s Happening?**

1. **LocalStack CLI Detected Docker Mode:**
   [11:20:31] starting LocalStack in Docker mode confirms that LocalStack is trying to run using Docker.
2. **Container Image Not Found Locally:**
   [11:25:13] container image not found on host indicates that Docker is downloading the localstack/localstack image from Docker Hub since it’s not found locally.

3. **Pulling the Container Image:**
   ⁝ Pulling container image localstack/localstack confirms that the image download process is ongoing.

**What to Expect Next**

If the internet connection is stable, Docker should complete the download in a few minutes. After the image is downloaded, LocalStack will initialize, and the services (e.g., S3, DynamoDB) will start. This will be confirmed by further logs.

**How to Verify It's Working**

1. After the image download is complete, the user can check if the container is running with the following command:
   docker ps
   The container should be listed with the name localstack.
2. To verify that LocalStack is running, the endpoint should be accessed:
   curl http://localhost:4566
   A response should be returned, confirming that LocalStack is active.

## Steps to Set Up a Virtual Machine in LocalStack:

**1. Simulate EC2 Service**

LocalStack emulates a limited set of EC2 functionalities. The goal is to create mock resources like key pairs, security groups, and instances.

**2. Create a Key Pair**

Use the AWS CLI to generate a key pair:

***aws ec2 create-key-pair --key-name local-key --endpoint-url=%AWS_ENDPOINT_URL%***

The output will include the generated public/private key pair.

**3. Create a Security Group**

Create a security group to define network rules:

***aws ec2 create-security-group --group-name local-sg --description "Local Security Group" --endpoint-url=%AWS_ENDPOINT_URL%***

**4. Run an Instance**

Launch a mock EC2 instance using:

***aws ec2 run-instances --image-id ami-12345678 --count 1 --instance-type t2.micro --key-name local-key --security-group-ids sg-12345678 --endpoint-url=%AWS_ENDPOINT_URL%***

Replace ami-12345678 with an example AMI ID that is recognized by LocalStack.

**Example Output:**

Security Group ID: sg-2cd410ccd533c7f8b

Image ID: ami-a2678d778fc6

**Command:**

***aws ec2 run-instances --image-id ami-a2678d778fc6 --count 1 --instance-type t2.micro --key-name local-key --security-group-ids sg-2cd410ccd533c7f8b --endpoint-url=%AWS_ENDPOINT_URL%***

**What Happens When You Run This Command?**

- AWS CLI creates a single EC2 instance based on ami-a2678d778fc6.
- The instance is t2.micro, suitable for low-resource tasks.
- Uses local-key key pair for SSH access.
- sg-2cd410ccd533c7f8b security group manages traffic.
- Endpoint URL directs requests to the specified AWS service.

**Example Use Case**

Set up a local test server (e.g., Ubuntu) with a custom AWS endpoint (http://localhost:4566) and controlled security settings.

**5. List Instances**

Verify the instance creation:

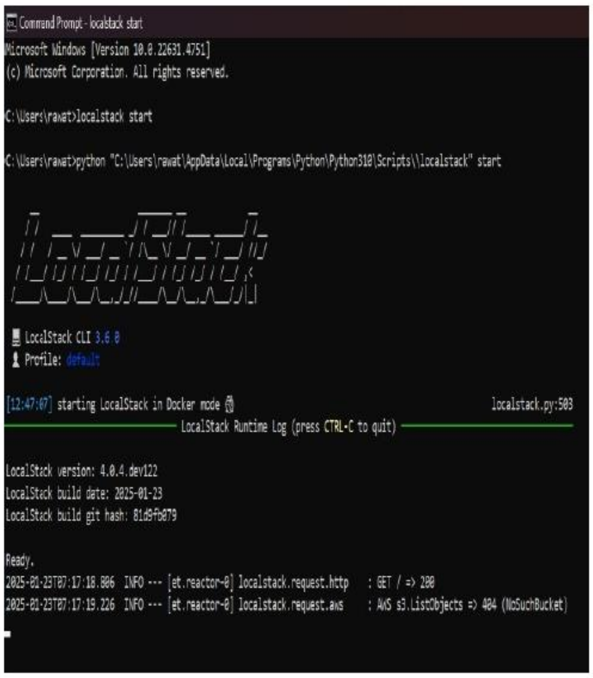
***aws ec2 describe-instances --endpoint-url=%AWS_ENDPOINT_URL%***

## Conclusion:

In this experiment, we have successfully executed the necessary steps to set up and simulate an EC2 instance using LocalStack. We created a key pair, defined a security group, launched a mock instance, and verified its status using AWS CLI commands. The output for all the executed commands is provided on the next page for reference.

## *Experiment 1 Output*

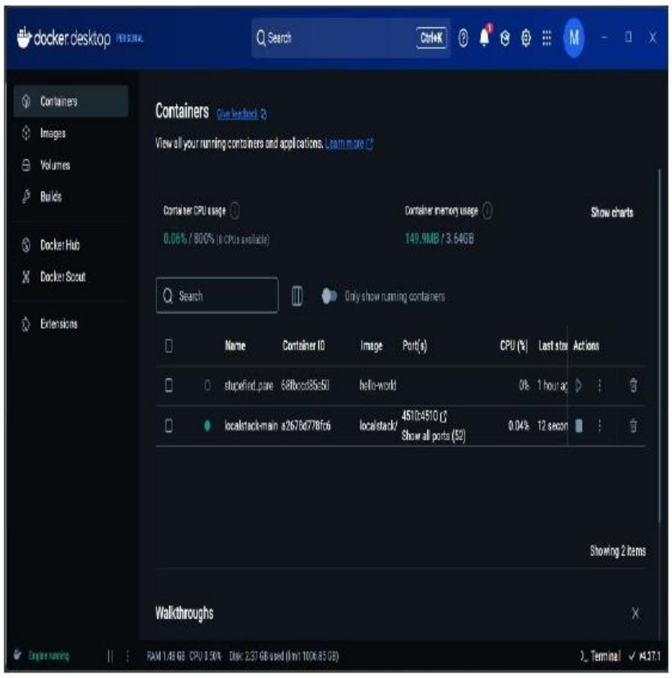
### *Running AWS using Localstack*



```
Command Prompt - localstack start
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rawat>localstack start

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

 LocalStack CLI 3.6.0
 Profile: default

[12:47:07] starting LocalStack in Docker mode                                    localstack.py:503
──────────────── LocalStack Runtime Log (press CTRL-C to quit) ────────────────

LocalStack version: 4.0.4.dev122
LocalStack build date: 2025-01-23
LocalStack build git hash: 81d9fb079

Ready.
2025-01-23T07:17:18.806  INFO --- [et.reactor-0] localstack.request.http   : GET / => 200
2025-01-23T07:17:19.226  INFO --- [et.reactor-0] localstack.request.aws    : AWS s3.ListObjects => 404 (NoSuchBucket)
```

**Fig 1: Starting Localstack in the Console**

### *Running Docker*



Containers

View all your running containers and applications.

Container CPU usage: 0.06% / 800% (8 CPUs available)

Container memory usage: 149.9MB / 3.64GB

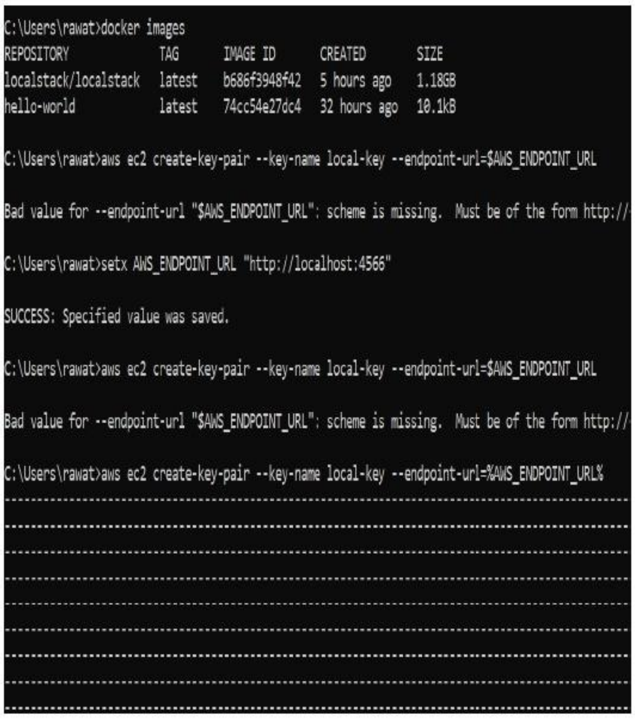
|  | Name | Container ID | Image | Port(s) | CPU (%) | Last star |
|---|---|---|---|---|---|---|
|  | stupefied_pare | 68fbccd65e50 | hello-world |  | 0% | 1 hour ag |
| ● | localstack-main | a2678d778fc6 | localstack/ | 4510:4510 Show all ports (52) | 0.04% | 12 secon |

Showing 2 items

RAM 1.48 GB CPU 0.50% Disk: 2.37 GB used (limit 1006.85 GB)

*Fig 2: Docker for Desktop in windows*

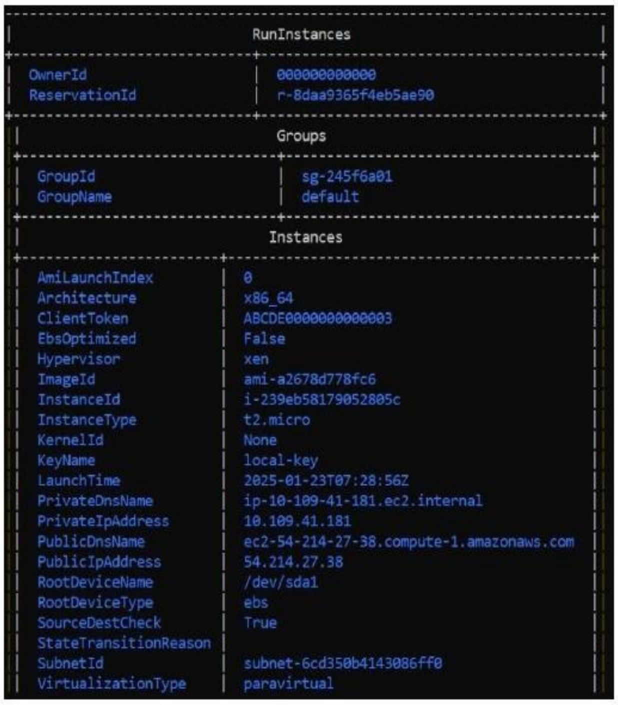
### *Creating a New Instance*



```
C:\Users\rawat>docker images
REPOSITORY             TAG       IMAGE ID       CREATED         SIZE
localstack/localstack  latest    b686f3948f42   5 hours ago     1.18GB
hello-world            latest    74cc54e27dc4   32 hours ago    10.1kB

C:\Users\rawat>aws ec2 create-key-pair --key-name local-key --endpoint-url=$AWS_ENDPOINT_URL

Bad value for --endpoint-url "$AWS_ENDPOINT_URL": scheme is missing.  Must be of the form http://

C:\Users\rawat>setx AWS_ENDPOINT_URL "http://localhost:4566"

SUCCESS: Specified value was saved.

C:\Users\rawat>aws ec2 create-key-pair --key-name local-key --endpoint-url=$AWS_ENDPOINT_URL

Bad value for --endpoint-url "$AWS_ENDPOINT_URL": scheme is missing.  Must be of the form http://

C:\Users\rawat>aws ec2 create-key-pair --key-name local-key --endpoint-url=%AWS_ENDPOINT_URL%
-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------
```

*Fig 3: Docker Images and New Instances*

### *Starting Virtual Machine*



```
-----------------------------------------------------------------------
|                            RunInstances                             |
+-----------------------------------+---------------------------------+
|  OwnerId                          |  000000000000                   |
|  ReservationId                    |  r-8daa9365f4eb5ae90            |
+-----------------------------------+---------------------------------+
||                              Groups                               ||
|+-----------------------------------+-------------------------------+|
||  GroupId                          |  sg-245f6a01                  ||
||  GroupName                        |  default                      ||
|+-----------------------------------+-------------------------------+|
||                             Instances                             ||
|+--------------------------+----------------------------------------+|
||  AmiLaunchIndex          |  0                                     ||
||  Architecture            |  x86_64                                ||
||  ClientToken             |  ABCDE0000000000003                    ||
||  EbsOptimized            |  False                                 ||
||  Hypervisor              |  xen                                   ||
||  ImageId                 |  ami-a2678d778fc6                      ||
||  InstanceId              |  i-239eb58179052805c                   ||
||  InstanceType            |  t2.micro                              ||
||  KernelId                |  None                                  ||
||  KeyName                 |  local-key                             ||
||  LaunchTime              |  2025-01-23T07:28:56Z                  ||
||  PrivateDnsName          |  ip-10-109-41-181.ec2.internal         ||
||  PrivateIpAddress        |  10.109.41.181                         ||
||  PublicDnsName           |  ec2-54-214-27-38.compute-1.amazonaws.com ||
||  PublicIpAddress         |  54.214.27.38                          ||
||  RootDeviceName          |  /dev/sda1                             ||
||  RootDeviceType          |  ebs                                   ||
||  SourceDestCheck         |  True                                  ||
||  StateTransitionReason   |                                        ||
||  SubnetId                |  subnet-6cd350b4143086ff0              ||
||  VirtualizationType      |  paravirtual                           ||
```

*Fig 4: Virtual Machine & Working*

# Resources

1. **GitHub Repository: Cloud Computing**
   This repository focuses on cloud computing and demonstrates how to set up virtual machines, S3, and other services using LocalStack. It provides a comprehensive guide to simulating AWS services locally for development and testing purposes. Additionally, it contains detailed documentation for every experiment conducted.

2. **LocalStack Official Documentation**
   This resource provides in-depth guidance on setting up and using LocalStack to emulate AWS cloud services for development and testing.

3. **AWS Documentation**
   The official AWS documentation offers detailed explanations, best practices, and tutorials for working with various AWS services, including EC2, S3, Lambda, and more.

4. **Cloud Computing Concepts - Coursera**
   A well-structured online course that covers fundamental cloud computing concepts, including virtualization, networking, and cloud service models like IaaS, PaaS, and SaaS.