**SRM Institute of Science and Technology**
**Delhi – Meerut Road, Sikri Kalan, Ghaziabad, Uttar Pradesh – 201204**
**Department of Computer Applications**
**Circular – 2023-24**
**BCA DS 5th Sem**

**Introduction to Cloud Computing (UDS23D02J)**

# Lab Manual

## Lab 1: Create a Virtual Machine

**Title:** Creating a Virtual Machine in [Cloud Provider Name]

**Aim:** To create and configure a virtual machine (VM) instance on a cloud platform.

**Procedure:**

1. Log in to your [Cloud Provider Name] account (e.g., AWS, Azure, Google Cloud).
2. Navigate to the Virtual Machines service.
3. Click "Create" or "Add" to start the VM creation process.
4. Choose an operating system image (e.g., Ubuntu, Windows Server).
5. Select a VM size/type (e.g., number of CPUs, memory).
6. Configure storage (e.g., disk size, type).
7. Set up networking (e.g., virtual network, subnet, public IP address).
8. Configure security settings (e.g., security groups, firewall rules).
9. Review the configuration and click "Create" to launch the VM.
10. Connect to the VM using SSH (for Linux) or Remote Desktop Protocol (RDP) (for Windows).

**Source Code:** (This is more of a configuration process than writing code. However, cloud providers often provide command-line tools or SDKs. Here's an example using the AWS CLI):

```
# Example using AWS CLI (replace with your specific parameters)
aws ec2 run-instances \
  --image-id ami-xxxxxxxxxxxxxxxx \
  --instance-type t2.micro \
  --key-name my-key-pair \
  --security-group-ids sg-xxxxxxxxxxxxxxxx \
  --subnet-id subnet-xxxxxxxxxxxxxxxx
```

**Input:**

11. Cloud provider credentials (account ID, access keys)
12. Operating system image selection
13. VM size/type selection
14. Storage configuration
15. Network configuration
16. Security group rules

**Expected Output:**

17. A running virtual machine instance.
18. The ability to connect to the VM via SSH or RDP.
19. Verification of the VM's operating system and resources.

**Lab 2: Installation of Platforms**

**Title:** Installing [Platform Name] on a Virtual Machine

**Aim:** To install a specific platform (e.g., Docker, Kubernetes, Hadoop) on a cloud-based virtual machine.

**Procedure:**

1. Create a VM (as in Lab 1).
2. Connect to the VM via SSH/RDP.
3. Follow the installation instructions for the chosen platform (these will vary significantly). This usually involves:
    - Updating system packages.
    - Downloading and installing the platform software.
    - Configuring the platform's services.
    - Starting the platform's services.
4. Verify the installation by checking the platform's status or running a simple test command.

**Source Code:** (This will be a series of shell commands or a setup script, depending on the platform)

**Input:** VM details, platform-specific installation files/URLs.

**Expected Output:** Successful installation of the platform, verified services.

**Lab 3: Deploying Existing Apps**

**Title:** Deploying an Application on [Platform Name]

**Aim:** To deploy a pre-existing application to a cloud platform.

**Procedure:**

1. Prepare the application for deployment (e.g., package it into a Docker container, create a deployment package).
2. Transfer the application package to the VM or a suitable storage location.
3. Deploy the application using the platform's deployment tools (e.g., docker run, kubectl apply).
4. Configure any necessary environment variables or dependencies.
5. Access the application through its URL or IP address.

**Source Code:** (Deployment scripts, Dockerfiles, Kubernetes YAML files)

**Input:** Application code, platform-specific deployment configurations.

**Expected Output:** Running application, accessible via a network.

**Lab 4: Create a Drop Box using Google API**

**Title:** Creating a Drop Box Application using Google Drive API

**Aim:** To develop a web application that allows users to upload files to a designated folder in Google Drive.

**Procedure:**

1. Set up a Google Cloud Project and enable the Google Drive API.
2. Create credentials (client ID and client secret) for your application.
3. Develop the application (using a language like Python, JavaScript, etc.) to:
    - Authenticate users using OAuth 2.0.
    - Provide a user interface for file selection.
    - Use the Google Drive API to upload selected files to a specific folder.
4. Deploy the application to a web server or cloud hosting platform.

**Source Code:** (Python, JavaScript, HTML)

**Input:** Google API credentials, files to upload.

**Expected Output:** Files uploaded to a specific Google Drive folder.

**Lab 5: Transfer Data using Google APPS**

**Title:** Transferring Data with Google Drive API

**Aim:** To create an application that transfers data (files) between locations using the Google Drive API.

**Procedure:**

1. Set up a Google Cloud Project and enable the Google Drive API.
2. Obtain necessary credentials.
3. Develop an application to:

    Authenticate users.

    List files in a source location.

    Download files from the source.

    Upload files to a destination location within Google Drive.

**Source Code:** (Python, JavaScript)

**Input:** Google Drive API credentials, source and destination locations.

**Expected Output:** Data transferred between specified locations in Google Drive.

**Lab 6: Upload and Download using Google APPS**

**Title:** File Upload and Download with Google Drive API

**Aim:** To build an application that uploads and downloads files to and from Google Drive.

**Procedure:**

1. Enable the Google Drive API.
2. Get credentials.
3. Develop the application:
    - Implement file upload functionality.
    - Implement file download functionality.
    - Provide a user interface for selecting files.
4. Deploy the application.

**Source Code:** (Python, JavaScript)

**Input:** Google Drive API credentials, files for upload/download.

**Expected Output:** Successful file uploads and downloads to/from Google Drive.

## Lab 7: Encryption and Decryption of Text

**Title:** Text Encryption and Decryption

**Aim:** To implement a text encryption and decryption application.

**Procedure:**

1. Choose an encryption algorithm (e.g., AES, RSA).
2. Develop an application (in any language) to:

   Generate or obtain encryption keys.

   Encrypt user-provided text using the chosen algorithm and key.

   Decrypt the encrypted text using the corresponding key.

3. Provide a user interface for inputting text and keys.

**Source Code:** (Python, Java, etc. Libraries like cryptography in Python, javax.crypto in Java)

**Input:** Text to encrypt/decrypt, encryption key.

**Expected Output:** Correctly encrypted and decrypted text.

**Lab 8: Simple Experiments in Cloud Sim**

**Title:** CloudSim Simulation Experiments

**Aim:** To conduct simulation experiments using the CloudSim simulator.

**Procedure:**

1. Install CloudSim.
2. Design simulation scenarios, including:
    - Cloud data center configurations.
    - Virtual machine provisioning policies.
    - Workload models.
    - Resource allocation strategies.
3. Write CloudSim code to implement the scenarios.
4. Run the simulations and collect results (e.g., execution time, resource utilization).
5. Analyze the results and draw conclusions.

**Source Code:** (Java code using the CloudSim library)

**Input:** Simulation parameters, CloudSim configuration files.

**Expected Output:** Simulation results, performance metrics.

## Lab 9: Develop a Hello World application using Google App Engine

**Title:** "Hello, World!" on Google App Engine

**Aim:** To deploy a simple "Hello, World!" application on Google App Engine.

**Procedure:** 1. Set up a Google Cloud Project and enable the App Engine API. 2. Create a "Hello, World!" application (e.g., in Python, Java, Go, Node.js, PHP, Ruby). 3. Create the necessary configuration files (e.g., app.yaml). 4. Deploy the application to Google App Engine using the gcloud command-line tool. 5. Access the application through the provided URL.

**Source Code:** (Python, Java, etc.)

**Input:** Google Cloud project details.

**Expected Output:** A web page displaying "Hello, World!".

## Lab 10: Develop a Guestbook Application using Google App Engine

**Title:** Guestbook Application on Google App Engine

**Aim:** To develop and deploy a guestbook web application on Google App Engine, allowing users to leave messages.

**Procedure:**

1. Set up a Google Cloud Project and enable the App Engine API.
2. Develop the guestbook application, including:

   A data model to store guestbook entries.

   A web page to display entries and a form to submit new entries.

   Server-side code to handle form submissions and store data.

3. Create configuration files.
4. Deploy to Google App Engine.

**Source Code:** (Python, Java)

**Input:** Google Cloud project details, guestbook entries.

**Expected Output:** A web application where users can view and add guestbook entries.

## Lab 11: Develop a Windows Azure Hello World application

**Title:** "Hello, World!" on Azure App Service

**Aim:** To deploy a "Hello, World!" application on Microsoft Azure App Service.

**Procedure:** 1. Create an Azure account. 2. Create an App Service web app. 3. Develop a "Hello, World!" application (e.g., in .NET, Python, Node.js). 4. Deploy the application to the Azure App Service. 5. Access the application via the provided URL.

**Source Code:** (.NET, Python, etc.)

**Input:** Azure account details.

**Expected Output:** A web page displaying "Hello, World!".

## Lab 12: Create a Warehouse Application in SalesForce.com

**Title:** Warehouse Application in Salesforce

**Aim:** To create a warehouse management application in Salesforce, including objects, fields, and potentially Apex code.

**Procedure:**

1. Create a Salesforce Developer Edition account.
2. Define custom objects (e.g., Warehouse, Product, Inventory).
3. Define fields for the objects (e.g., Warehouse Name, Product Name, Quantity).
4. Create page layouts and tabs for the objects.
5. (Lab 14) Write Apex code to implement business logic (e.g., inventory management, reporting).
6. Create Visualforce pages or Lightning components for the user interface.

**Source Code:** (Salesforce declarative configuration, Apex code, Visualforce/Lightning code)

**Input:** Salesforce Developer Edition account details, warehouse data.

**Expected Output:** A functional warehouse management application in Salesforce.

**Lab 13: Implementation of SOAP Web Services**

**Title:** Implementing SOAP Web Services

**Aim:** To create and consume a SOAP web service.

**Procedure:**

1. Choose a platform (e.g., Java, .NET) to create the web service.
2. Define the web service interface (WSDL).
3. Implement the web service logic.
4. Deploy the web service to a server.
5. Create a client application (in any language) to consume the web service.
6. Send SOAP requests from the client to the service and process the responses.

**Source Code:** (Java, .NET, WSDL, client-side code)

**Input:** Data for SOAP requests.

**Expected Output:** Successful communication between the SOAP client and web service.