Cloud Computing

Slip 1

Create and host static web page using any cloud provider

ANS:

Creating and hosting a static web page using a cloud provider involves a few steps (for 20 ,marks)

Step 1: Create an Amazon S3 Bucket

Sign in to AWS Console:

Go to the AWS Management Console -> sign in to your account.

Navigate to S3:

Find and select the "S3" service from the AWS Management Console.

Create a Bucket:

- Click on the "Create bucket" button->Choose a globally unique name for your bucket->Select the region for your bucket->Click "Create bucket."

Step 2: Upload Your Web Page Files

Select your Bucket:

- Click on the newly created bucket->Navigate to the "Objects" tab.

Upload Files:

- Click "Upload->Add your HTML, CSS, and other static files.
- Click "Next" through the remaining steps, and then click "Upload."

Step 3: Configure Bucket for Static Website Hosting

Navigate to Properties:

- In the bucket view, go to the "Properties" tab.

Enable Static Website Hosting:

- Click on "Static website hosting->Choose "Use this bucket to host a website->Set the index document (e.g., index.html).

Save Changes:

- Click "Save changes."

Step 4: Make Your Bucket Public

Bucket Policy (Optional):

- Go to the "Permissions" tab->Update the bucket policy to make your bucket publicly accessible.

- Step 5: Access Your Website

Find Endpoint:

- In the "Static website hosting" section, there is an "Endpoint" URL. This is the URL where your website is hosted.

Access Your Website:

- Open a web browser and enter the endpoint URL->Your static website should now be accessible.

You've successfully created and hosted a static web page using Amazon S3.

❖ **for(10 marks)**

To create a static web page using any cloud provider:

Choose a Cloud Storage Service:

- Use a cloud storage service like AWS S3, Google Cloud Storage, or Azure Blob Storage.

Upload HTML, CSS, and JS Files:

- Upload your static files (HTML, CSS, JS) to the cloud storage.

Configure Hosting:

- Set the storage container or bucket to be publicly accessible.

- Enable static website hosting if available.

Access the Web Page:

- Obtain the public URL provided by the cloud storage to access your static web page.

Slip:4

Q.1) Working and Implementation of Software as a Service(google)

Ans:

(Write what is Saas,HOW IT WORKS,HOW ITS IMPLEMENTED then write following Steps)

Implementing Software as a Service (SaaS) on a platform like Google Cloud involves utilizing various services provided by Google.

creating a web-based task manager using Google Cloud services, including Google App Engine (GAE) for hosting, Cloud Firestore for the database, and Firebase Authentication for user authentication.

1. Set Up Your Project

Google Cloud Console:

Enable APIs and Services:

- In your project, enable the App Engine, Cloud Firestore, and Firebase Authentication APIs.

Set Up Firebase:

- In the Firebase console, create a new project.
- Enable Firebase Authentication and Firestore.

2. Write the Code

3. Deploy to Google App Engine

Make sure you have the google cloud SDK installed.
Open a terminal and navigate to your project folder.
Run the following commands:
gcloud app deploy

4. Access Your SaaS Application

Once the deployment is successful, you can access your SaaS task manager using the provided App Engine URL.

Slip 5:

Q.1 Working and implementation of Infrastructure as a service

Ans:

(First write what is Iaas,how it works then write steps)
Steps:

Provision Virtual Resources:

- Users access virtualized computing resources (servers, storage, networking) over the internet.

Cloud Management Console:

- Resources are provisioned, configured, and managed through a cloud provider's web-based dashboard.

Scalability:

- IaaS allows dynamic scaling of resources, adjusting capacity

based on demand.

Pay-as-You-Go Model:

- Users pay for the resources consumed, avoiding upfront hardware investments.

Example Providers:

- AWS EC2, Azure Virtual Machines, Google Compute Engine.

Use Cases:

Ideal for businesses needing flexible and scalable computing infrastructure without the burden of physical hardware

.

Slip 6,slip 13,slip14

Q.1 Write program for web feed.

Ans:

```
from flask import Flask, render_template
from werkzeug.contrib.atom import AtomFeed
import datetime

app = Flask(__name__)

# Sample data
posts = [
    {
        'title': 'Post 1',
        'content': 'This is the content of post 1.',
```

```
        'date': datetime.datetime(2023, 1, 1),
    },
    {
        'title': 'Post 2',
        'content': 'This is the content of post 2.',
        'date': datetime.datetime(2023, 2, 1),
    },
]

# Route to generate the feed
@app.route('/feed')
def feed():
    feed=AtomFeed('MyBlog Feed', feed_url='https://yourblog.com/feed',
url='https://yourblog.com')

    for post in posts:
        feed.add(post['title'], unicode(post['content']),
                 content_type='html',
                 author='Your Name',
                 url='https://yourblog.com/post/{}'.format(post['title']),
                 updated=post['date'],
                 published=post['date'])

    return feed.get_response()

# Route for the home page
@app.route('/')
def home():
    return render_template('index.html', posts=posts)

if __name__ == '__main__':
    app.run(debug=True)
```

Slip:9

Q.1 Create Virtual Machine using Virtual box

Ans:

Install VirtualBox:

- Download and install VirtualBox from the official website.

Open VirtualBox:

- Launch VirtualBox and click "New" to create a new virtual machine.

Configure VM Settings:

- Name your VM, choose the type and version of the operating system.

Allocate Memory:

- Assign the amount of RAM for the VM.

Create Virtual Hard Disk:

- Create a virtual hard disk with a specified size.

Install Operating System:

- Mount the OS ISO file, start the VM, and install the operating system.

Complete Setup:

- Follow on-screen instructions to finish the OS installation.

Run Virtual Machine:

- Start the VM and use it as a standalone computer within your host machine.

Slip 12 .

Demonstrate how to managing cloud computing resourses.

Ans:

Managing cloud computing resources involves provisioning, monitoring, scaling, and optimizing cloud resources to meet your application's requirements efficiently.

1. Sign Up for a Cloud Provider

2. Access the Cloud Console

3. Provisioning Resources

Create a Virtual Machine (EC2 in AWS):

Create a Database (RDS in AWS):

4. Monitoring Resources

Set Up CloudWatch (AWS):

In AWS, navigate to the CloudWatch dashboard.

Create alarms to monitor your resources. For example, set up an alarm to notify you if CPU utilization of an EC2 instance goes above a certain threshold.

5. Scaling Resources

6. Managing Storage

Create and Manage Buckets (S3 in AWS):

7. Networking

Create a Virtual Private Cloud (VPC in AWS):

8. Security and Identity Management

9. Cost Management

10. Resource Cleanup

Deallocate Resources:

When you are finished with resources, ensure to terminate or delete them to avoid unnecessary charges.

Slip 20,slip 25

Create Virual Machine and perform basic shell operations

Ans:

Write Installation steps then write Shell operations

Shell Opeartion	Command
Check System Information:	uname -a
List Files in the Current Directory	ls
Change Directory:	cd your_directory_path
Create a Directory:	mkdir new_directory
Copy a File:	cp source_file destination
Move/Rename a File:	mv old_file new_file
Remove/Delete a File:	rm file_to_remove

Print Working Directory:	pwd
Display File Content:	cat file_name
Edit a File:	nano file_name
Exit the Terminal:	exit

Slip 21:

Show practical implementation of cloud on single sign on.

Ans:

Choose Identity Provider (IdP):

- Sign up with an IdP like Auth0, Okta, or Azure AD.

Create an Application:

- Set up a new application within the IdP for your cloud-based service.

Configure SSO Settings:

- Define callback URLs, logout URLs, and other relevant settings.

Integrate IdP in Your App:

- Use IdP-provided SDKs or libraries to integrate SSO

functionality into your application.

Implement SSO Buttons:

- Add buttons or links in your app to initiate the SSO login process.

User Authentication:

- Users authenticate once with the IdP, and subsequent logins to other connected services are automatic.

User Management:

- Leverage IdP features for user management, role assignments, and security policies.

Run and Test:

- Host and run your application, testing SSO functionality with different user accounts.

Implementing SSO simplifies user authentication across multiple services, enhancing user experience and security in a cloud environment.

Slip 23

Practical Implementation of File Sharing and storage as a service.

Ans:

Choose Cloud Storage Provider:

- Select a provider like AWS S3, Google Cloud Storage, or Microsoft Azure Blob Storage.

Create Storage Container/Bucket:

- Set up a container or bucket in the chosen storage service to hold your files.

Configure Access Permissions:

- Define access controls, making the storage publicly accessible or restricted based on requirements.

Upload Files:

- Use the storage provider's interface or API to upload files to

the designated container or bucket.

Generate Access Links:

- Generate secure links for shared files, allowing controlled access.

Implement Authentication:

- For controlled access, integrate identity and access management features provided by the storage service.

Monitor and Manage Files:

- Utilize the storage provider's dashboard or API to monitor usage, manage files, and adjust storage settings.

Example Use Cases:

- Ideal for collaborative work, backups, and sharing large datasets.

Implementing file sharing and storage as a service involves selecting a cloud storage provider, configuring access, uploading files, and managing storage through the provider's tools or APIs.

Slip 26:

Working of Software as a Service (SaaS) on Amazon:

Ans:

Infrastructure Services:

- **Amazon Elastic Compute Cloud (EC2):** SaaS providers can use EC2 instances to host their application servers. EC2 provides scalable computing capacity in the cloud.

- **Amazon Simple Storage Service (S3):** SaaS providers can store and retrieve data using S3. This is particularly useful for storing static assets, such as images or documents.

Database Services:

- **Amazon Relational Database Service (RDS):** SaaS applications often require a relational database. RDS makes it easy to set up, operate, and scale a relational database in the cloud. MySQL, PostgreSQL, and other database engines are supported.
- **Amazon DynamoDB:** For NoSQL database requirements, DynamoDB is a fully managed service that provides fast and predictable performance with seamless scalability.

Scalability and Load Balancing:

- **Amazon Elastic Load Balancing (ELB):** ELB distributes incoming application traffic across multiple targets, such as EC2 instances, to ensure the application's availability and fault tolerance.

Identity and Access Management:

- **AWS Identity and Access Management (IAM):** IAM allows SaaS providers to manage access to AWS services securely. It enables the creation of users and groups and the assignment of permissions.

Networking:

- **Amazon Virtual Private Cloud (VPC):** VPC allows users to provision a logically isolated section of the AWS Cloud where they can launch AWS resources in a virtual

network.

Monitoring and Logging:

- **Amazon CloudWatch:** CloudWatch provides monitoring for AWS resources and applications in real-time. SaaS providers can use it to collect and track metrics, collect and monitor log files, and set alarms.

Application Deployment and Management:

- **AWS Elastic Beanstalk:** Elastic Beanstalk is a fully managed service that makes it easy to deploy and run applications in multiple languages. It abstracts the underlying infrastructure, allowing developers to focus on writing code.

Content Delivery:

- **Amazon CloudFront:** CloudFront is a content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally.

Serverless Computing:

- **AWS Lambda:** SaaS providers can use Lambda for serverless computing. It allows running code without provisioning or managing servers.

0. **Security:**

- **AWS Key Management Service (KMS):** KMS enables the creation and control of encryption keys that can be used to encrypt data