ST. ANNE'S COLLEGE OF ENGINEERING AND TECHNOLOGY
(Approved by AICTE, New Delhi. Affiliated to Anna University, Chennai)
Accredited by NAAC
ANGUCHETTYPALAYAM, PANRUTI – 607 106.

LAB MANUAL

CCS336 – CLOUD SERVICE MANAGEMENT LABORATORY

Regulation 2021

Year / Semester : III / VI

PREPARED BY

Ms. K. KAYALVIZHI, M.E.,
Assistant Professor / CSE Dept

EX.NO:01	Create a Cloud Organization in AWS/Google Cloud/or any equivalent Open-Source cloud software like OpenStack, Eucalyptus, Open Nebula with Role-based access control
Date:	

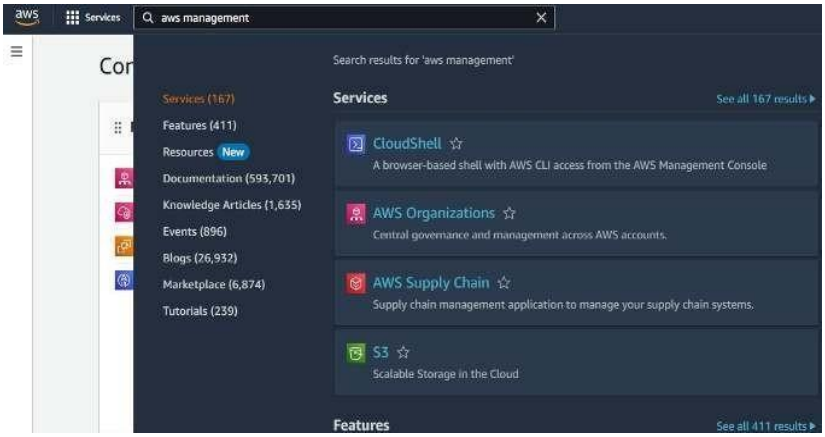
Aim:

To create a Cloud Organization in AWS with Roll-based access control.

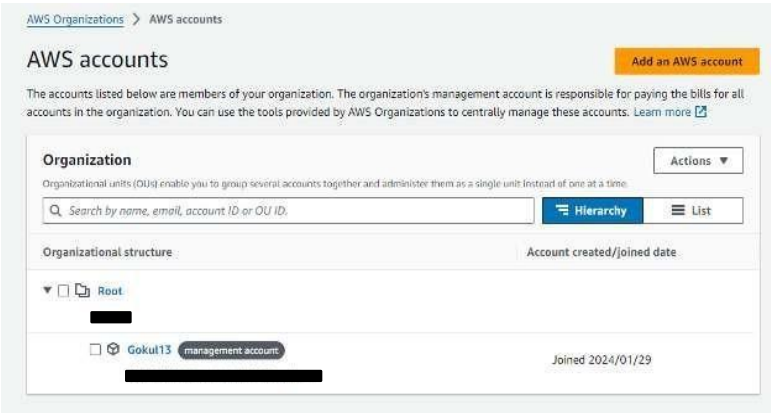
Procedure:

<u>To create an organization in AWS with role-based access, you can follow these general steps:</u>

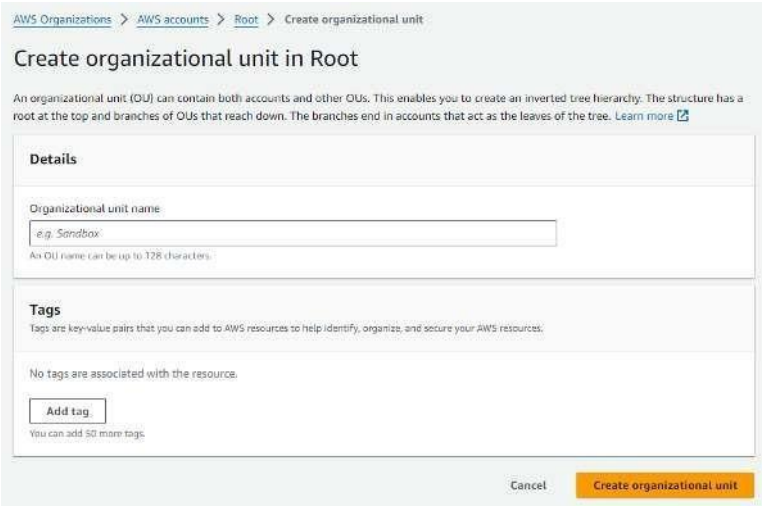
1. **Create an AWS account:** If you don't already have an AWS account, you'll need to create one. This will beyour management account and the root of your organization.



2. **Enable AWS Organizations:** From the AWS Management Console, navigate to the AWS Organizations service and enable it. This will create the organization with your management account as the master account.

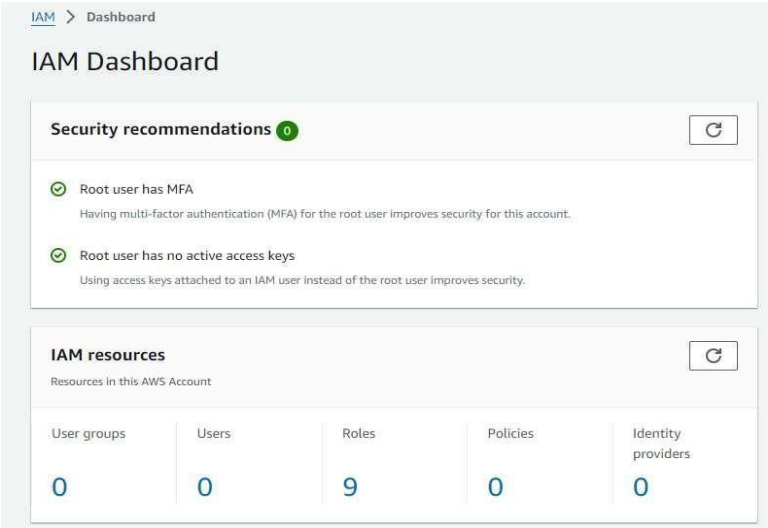


3. **Create OUs (Organizational Units):** You can create one or more OUs to organize your accounts. For example, you might create separate OUs for different departments or environments (e.g., production, staging, development).



4. **Create member accounts:** You can create new AWS accounts and invite existing accounts to join yourorganization as member accounts. You can add these accounts to the appropriate OUs.

5. **Create service control policies (SCPs):** SCPs are policies that you can attach to OUs or individual accountsto define the maximum set of actions that can be performed on resources in those OUs or accounts. This allows you to enforce role-based access and other security policies across your organization.

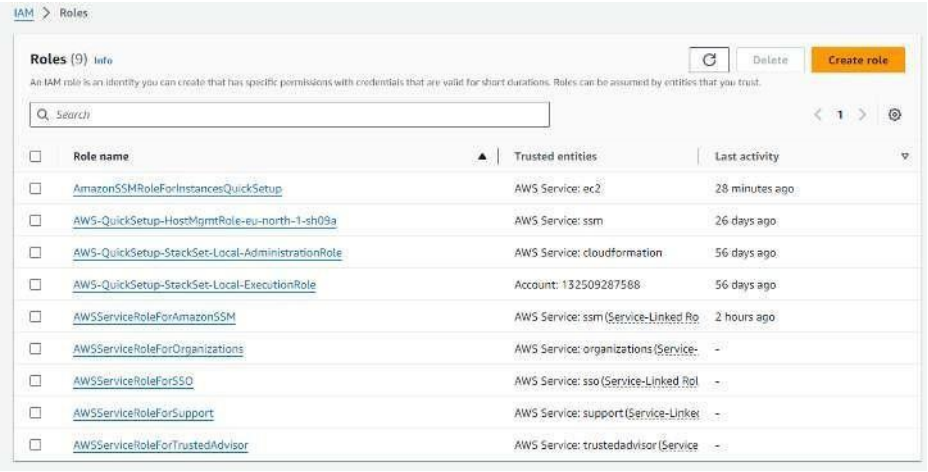


6. **Assign IAM roles:** You can create IAM roles in your management account and delegate specific permissions to them. You can then assume these roles from your member accounts to perform actions on resources in the management account or other member accounts.

7. **Configure permissions:** You can use IAM policies to control access to AWS services and resources. You can attach these policies to IAM users, groups, or roles in your management account or member accounts.

To create a role with specific permissions, you can follow these steps:

- Open the IAM console in your management account.
- Create a new role and choose the appropriate trusted entity (e.g., another AWS account, an AWS service, or your AWS Organizations).
- Define the permissions for the role by attaching an IAM policy or a service control policy (SCP).
- Save the role and note down the ARN (Amazon Resource Name) of the role.
- In the AWS Organizations console, attach the role to the appropriate OU or account.
- In the member account, assume the role to perform actions on resources in the management account or other member accounts.



Result:

Thus, the Cloud Organization was created in AWS with Role-Based Access Control was implemented successfully.

EX.NO:02	Create a Cost-model for a web application using various services and do Cost-benefit analysis
Date:	

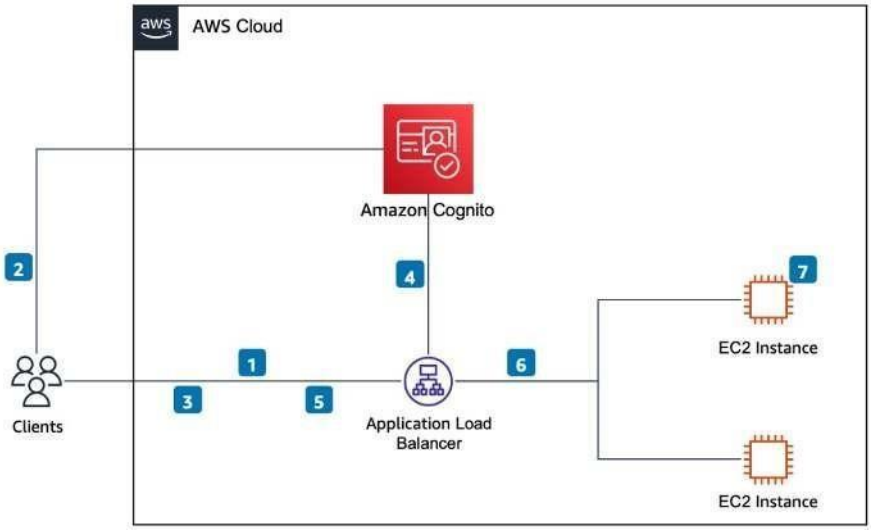
Aim:

To create a Cost-model for a web application using various services and make a analysis for Cost-benefit.

Procedure:

Creating a cost-model for a web application in AWS involves estimating the costs of using various AWS services for the application. Here's a general process to create a cost-model and do cost-benefit analysis:

1. **Identify the AWS services used by the web application:** Some common services used by web applications include Amazon S3, Amazon EC2, Amazon RDS, Amazon API Gateway, AWS Lambda,Amazon DynamoDB, Amazon CloudFront, and Amazon SNS.



2. **Estimate the costs of each service:** You can use the AWS Pricing Calculator to estimate the costs of each service. The pricing calculator allows you to enter the specific of your usage, such as the number of instances, storage size, and data transfer.
3. **Create a cost-model:** Once you have estimated the costs of each service, you can create a cost- model that summarizes the total costs. You can use a spreadsheet or a cloud cost management tool to create the cost-model.
4. **Do cost- benefit analysis:** After creating the cost-model, you can do a cost-benefit analysis to determine if the benefits of using AWS services outweigh the costs. You can compare the costs of using AWS services to the costs of running the application on-premises or using a different cloud provider.

Program:

Python code:

```
import boto3

# Create a session using your AWS
credentialssession = boto3.Session(
   aws_access_key_id='YOUR_ACCESS_KEY'
   aws_secret_access_key='YOUR_SECRET_K
   EY', region _ name='us-east-1'
)

# Create a Cost Explorer client cost
_explorer = session . client('c e')
# Define the time period for the cost-
modeltime _ period = {
   'Time                Unit':
   'MONTHS',      'Start':
   '2022-01-01',
   'End': '2022-12-31'
}

# Define the granularity of the cost-model
granularity = 'DAILY'

# Define the metrics for the cost-model
metrics = ['Blended Cost', 'UsageQuantity']

# Define the grouping for the cost-model
Group _ by = [{'Type': 'DIMENSION', 'Key': 'SERVICE'}]
```

```
# Get the cost and usage data
response = cost _ explorer . get _ cost _ and
   _usage        ( Time Period = time _ period,
   Granularity=granularity,
   Metrics=metrics,
   Group By = group
   _ by
)

# Print the cost and usage data
print(response)
```

Output:


```
{
   'Results By Time': [
      {
         'Time Period': {
            'Start': '2022-01-01',
            'End': '2022-12-31',
            'Time Unit': 'MONTHS'
         },
         'Groups': [
            {
               'Keys':          [
                  'AmazonEC2
                  '
               ],
               'Metrics':          {
                  'Blended Cost':
                  {
                     'Amount': '1234.56',
                     'Unit': 'USD'
                  },
```

'UsageQuantity':
'Amount': '1000.0',
'Unit': 'Hours'
}
}
},
{
'Keys': ['AWS Lambda'
],
'Metrics': {
'Blended Cost':
{
'Amount': '789.0',
'Unit': 'USD'
},
'UsageQuantity': {
'Amount': '5000000',
'Unit': 'requests'
}
}
}
]
}
],
'Response Meta data': {
'Request Id': 'abcdefg-1234-5678-90ab-cdefghijkl',
'HTTP Status Code': 200,
'HTTP Headers': {
'content-type': 'text/ xml ; charset=UTF-
8','content-length': '1234',

'date': 'Tue, 15 Feb 2022 12:34:56 GMT'
},
'Retry Attempts': 0
}
}

Result:

Thus, Cost-model for a web application using various services created and analysis was implemented successfully.

EX.NO:03	Create alerts for usage of Cloud Resources
Date:	

Aim:

To create alerts for usage of Cloud Resources.

Procedure:

To create alerts for usage of Cloud resources in AWS, you can use Amazon CloudWatch and AWS Lambda. Here's an example code that creates an alert for Amazon S3 bucket usage:

1. Create an IAM role for the Lambda function with the following policy.
2. Create a new Lambda function with the following code.
3. Set the Lambda function trigger to run every day at a specific time.
4. Create a CloudWatch alarm with the following code.

Program:

Policy for Role: (JSON code)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloud watch :Put Metric  Alarm",
        "cloud watch :Describe Alarms",
        "cloud watch :Get Metric Data",
        "cloud watch : Get Metric
        Statistics"
      ],
      "Resource": "*"
    },
    {
```

```
        "Effect":  "Allow",
        "Action": [
            "s3: Get Bucket Size"
        ],
        "Resource": [
            "arn : aws:s3:::your-bucket-name"
        ]
      }
   ]
}
```

New Lambda Function:

```
(Python)import boto3
import json

s3 = boto3.client('s3')
cloud watch = boto3.client('cloud watch')

def lambda _ handler (event, context):
   try:
      response = s3. head _ bucket (Bucket='your-bucket-
      name')size = response ['Content Length']

      cloud watch . put _ metric _ data(
         Namespace='S3',
         Metric Data= [
            {
               'Metric Name': 'Bucket
               Size','Dimensions': [
                  {
                     'Name': 'Bucket Name',
```

```
                'Value': 'your-bucket-name'
            },
        ],
        'Timestamp': datetime. date time.
        now(),'Value': size,
        'Unit': 'Bytes'
      },
    ]
  )
except Exception as e:
  print(e)
```

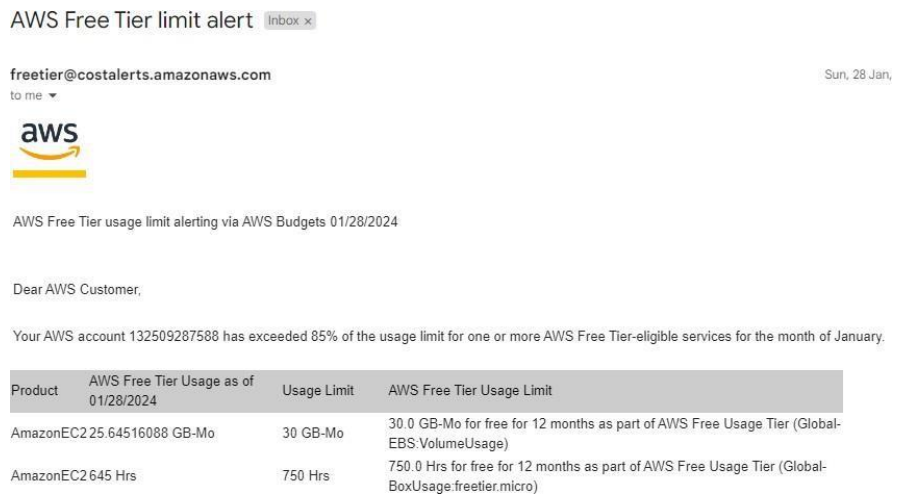
Cloud Watch Alarm:

```
(Python)import boto3
import datetime

cloud watch = boto3.client('cloud watch')

def create _ alarm ():
  try:
    cloud watch. Put _ metric _ alarm (Alarm
      Name='S3BucketSizeAlarm', Alarm Description='Alarm if
      S3 bucket size exceeds 10 GB',Namespace='S3',
      Metric Name='Bucket
      Size', Statistic='Sample
      Count', Period='86400',
      Evaluation Periods='1',
      Threshold='10000000000',
      Comparison Operator='Greater Than Threshold',
```

```
        Alarm Actions=[
            'arn:aws:sns:us-east-1:123456789012:your-sns-topic-arn'
        ],
        Dimensions=[
            {
                'Name':     'Bucket     Name',
                'Value': 'your-bucket-name'
            },
        ],
        Alarm Description='Alarm if S3 bucket size exceeds 10 GB'
    )
  except Exception as e:
    print(e)
```

create _ alarm ()

Output:



AWS Free Tier limit alert Inbox ×

freetier@costalerts.amazonaws.com Sun, 28 Jan,
to me

AWS Free Tier usage limit alerting via AWS Budgets 01/28/2024

Dear AWS Customer,

Your AWS account 132509287588 has exceeded 85% of the usage limit for one or more AWS Free Tier-eligible services for the month of January.

Product	AWS Free Tier Usage as of 01/28/2024	Usage Limit	AWS Free Tier Usage Limit
AmazonEC2	25.64516088 GB-Mo	30 GB-Mo	30.0 GB-Mo for free for 12 months as part of AWS Free Usage Tier (Global-EBS:VolumeUsage)
AmazonEC2	645 Hrs	750 Hrs	750.0 Hrs for free for 12 months as part of AWS Free Usage Tier (Global-BoxUsage:freetier.micro)

Result:

Thus, usage alerts for cloud resources were implemented successfully.

EX.NO:04	Create Billing alerts for your Cloud Organization
Date:	

Aim:

To create billing alerts for your Cloud Organization.

Procedure:

To create billing alerts for your Cloud Organization in AWS, you can follow these steps:

1. Sign in to the AWS Management Console and navigate to the Billing and Cost Management service.
2. In the navigation pane, choose "Budgets".
3. Click on "Create budget" and select "Cost budget".
4. Provide a name and description for your budget.
5. Choose the time period for your budget (e.g., Monthly, Quarterly, Annually).
6. Configure the budget threshold. You can choose to set a fixed budget amount or a percentage of your actual costs.
7. Configure the alerts. You can choose to receive alerts via email or Amazon SNS.

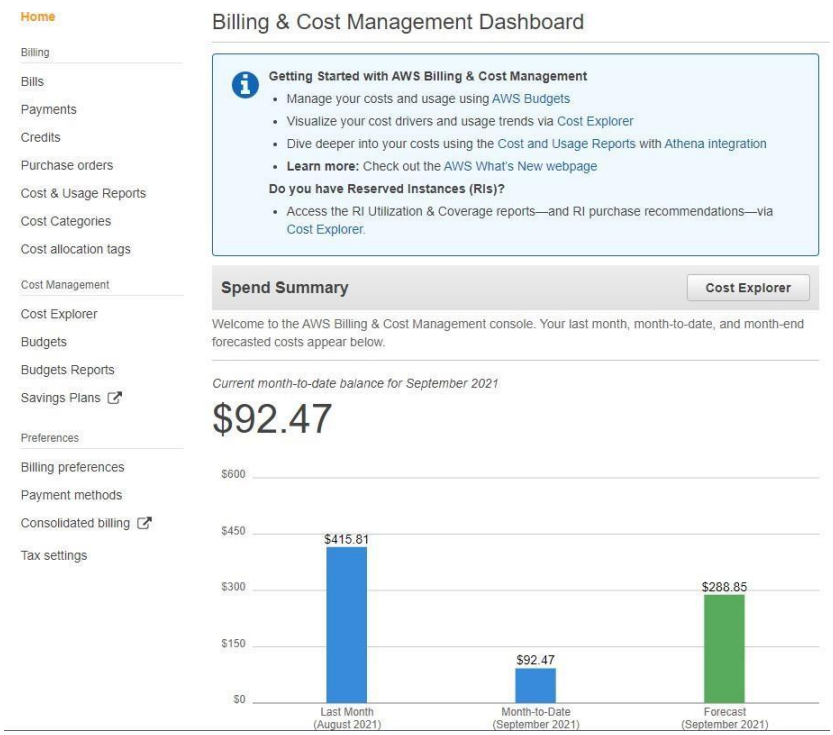
Program:

AWS CLI: (Bash)

```
aws budgets create-budget --account-id 123456789012 --budget \'{
 "Budget Name": "My Cost
 Budget","Budget
 Limit": {
 "Amount": "1000",
"Unit": "USD"
 },
 "Cost
 Filters": {
 "Linked
  Account": ["123456789012"]
 },
 "Cost Types": {
  "Include Tax":
  true,
```

"Include Subscription": true, "Use Blended": false, "Include Refund": true, "Include Credit": true, "Include Upfront": true, "Include Recurring": true, "Include Other Subscription": true, "Include Support": true, "Include Discount": true, "Use Amortized": false
},
"Time Unit": "MONTHLY",
"Budget Type": "COST",
"Notifications With Subscribers": [
{
"Notification": {
"Notification Type": "ACTUAL",
"Comparison Operator": "GREATER_THAN","Threshold": 100,
"Threshold Type": "PERCENTAGE", "Notification State": "ALARM"
},
"Subscribers": [
{
"Subscription Type": "EMAIL",
"Address": "you@example.com"
}
]
}
]
}'

Output:



Result:

Thus, billing alerts for your Cloud Organization were implemented successfully.

EX.NO:05	Compare Cloud cost for a simple web application across AWS, Azure and GCP and suggest thebest one
Date:	

Aim:

To compare Cloud cost for a simple web application across AWS, Azure and GCP and suggest the best one

Observation:

1. AWS: AWS offers a rich array of tools, including databases, analytics, management, IoT, security, and enterprise applications. AWS introduced per-second billing in 2017 for EC2 Linux-based instances andEBS volumes.

2. Azure: Azure has slightly surpassed AWS in the percentage of enterprises using it. Azure also offers various services for enterprises, and Microsoft's longstanding relationship with this segment makes it an easy choice for some customers. While Azure is the most expensive choice for general-purpose instances, it's one of the most cost-effective alternatives to compute-optimized instances.

3. Google Cloud Platform (GCP): GCP stands out thanks to its almost limitless internal research and expertise. GCP is different due to its role in developing various open-source technologies. Google Cloud is much cheaper than AWS and Azure for computing optimized cloud-based instances.

The best platform depends on your specific needs and requirements. If you need a wide array of tools and services, AWS might be the best choice. If you're looking for enterprise services and have a longstanding relationship with Microsoft, Azure could be your best bet.

Conclusion:

If you prioritize innovation and open-source technologies, GCP could be the right choice. For compute-optimized instances, GCP seems to be the most cost-effective. However, it's essential to understand your requirements fully before making a decision.

Result:

Thus, the comparison for Cloud cost for a simple web application across AWS, Azure and GCP were implemented successfully.