CLOUD SECURITY LAB **By-ShrestaMangala DSCE**

PLATFORM: Amazon Web Service

AWS Billing User Guide

AWS Billing and Cost Management provides a suite of features to help you set up your billing, retrieve and pay invoices, and analyse, organize, plan, and optimize your costs.

See the following overview of features to help you manage your cloud finances.

Features of AWS Billing and Cost Management

Topics

- Billing and payments
- Cost analysis
- Cost organization
- Budgeting and planning
- Savings and commitments

Billing and payments

Understand your monthly charges, view and pay invoices, and manage preferences for billing, invoices, tax, and payments.

Cost analysis

Analyse your costs, export detailed cost and usage data, and forecast your spending.

Cost organization

Organize your costs across teams, applications, or end customers.

Budgeting and planning

Estimate the cost of a planned workload, and create budgets to track and control costs.

Savings and commitments

Optimize resource usage and use flexible pricing models to lower your bill.

How to create a budget(User Guide) with alerts.

Step1: Click on Billing and cost management, choose Budget

Step2: Choose budget type. Click on Customize a budget to set parameters specific to your use case. You can customize the time period, the start month, and specific accounts.

Step3: Budget name, Provide a descriptive name for this budget.

Step4: Set budget amount with specific period, budgeting renewal method amount budget type.

Step5: Enter your budgeted amount

Step6: Budget scope. Click on All AWS services (Recommended).

Step7: Add on alert threshold with mail id (For one budget you can create up to 10 alerts).

Step8: Create budget-finish

Experiment-2

How to create billing and cost management on AWS Services.

Virtual Private Cloud (VPC) is a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network, allowing you to control your own IP address ranges, subnets, and security settings.

Subnet is a range of IP addresses within a Virtual Private Cloud (VPC) that you can use to launch AWS resources like EC2 instances, and they are restricted to a single Availability Zone

Route table contains a set of rules, called routes, that determine where network traffic from your subnet or gateway is directed, essentially acting as a map for routing within your Virtual Private Cloud (VPC).

Internet Gateway (IGW) is a horizontally scaled, redundant, and highly available VPC component that facilitates two-way communication between your VPC and the internet, enabling resources in public subnets with public IP addresses to connect to and receive connections from the internet.

Amazon RDS for SQL Server is a managed service that simplifies deploying, operating, and scaling SQL Server databases in the cloud, handling tasks like provisioning, patching, and backups, while you focus on application development.

Select a Region. For example North Virginia US -East

- Step1: Create VPC, Subnet, Route tables and Internet gateway.
- Step2: Create Relational database in AWS console application

Or

AWS pricing calculator

- Select RDS Custom for SQL Server instances
- Give value, database edition, Deployment option, licence, Pricing model
- View summary and Export estimated cost to pdf letterhead template. (Estimated cost).

Take a screenshot of Visual subnet calculator.

Take a screenshot of Export estimated cost to pdf letterhead template.

Experiment 3

How to Auto Stop Virtual Instances in AWS Cloud to Save the cost during non-working hours.

To automatically stop AWS EC2 instances during non-working hours, you can use tools like AWS Instance Scheduler or AWS Lambda functions. These solutions allow you to set schedules for starting and stopping instances, significantly reducing costs by ensuring they are not running when not needed.

Select a Region. For example North Virginia US -East

Step 1 Create a EC2 instance

Step 2 Create a IAM Role with Lambda services along with EC2Full access Permission

Step 3 Copy the Instance ID (Virtual Machine ID) from EC2 Instance

Step 4 Create a Lambda function along with the Code and replace the instance ID and region in the code.

```
import boto3

import botocore


def lambda_handler (event, context):

    ec2 = boto3.client('ec2', region_name='us-east-1')  # Corrected region name

    instance_id = "i-0cb771a7775f3909f"  # Replace with your EC2 Instance ID


    try:

        response = ec2.stop_instances(InstanceIds=[instance_id])

        return {

            'statusCode': 200,

            'body': f"Stopping instance {instance_id}. Response: {response}"

        }

    except botocore.exceptions.ClientError as e:

        return {
            'statusCode': 400,
            'body': f"Error stopping instance {instance_id}: {str(e)}"
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': f"An unexpected error occurred: {str(e)}"
        }
```

Step 5 Click on the Deploy button to save the code

Step 6 Go to the configuration option in Lambda function and change the execution time from 3 seconds to 1 min

Step 7 Click on the Test option in Lambda function and create a Test event

Step 8 Click on the Test

Step 9 Go to the EC2 console and check your EC2 instance is stopped or not

Take a screenshot of both EC2 instance START(Running) and EC2 instances STOPPED

EXPERIMENT 4

How to create AWS Services using Template.

Creating AWS services using templates typically involves using AWS CloudFormation, which allows you to define your infrastructure as code. This approach has several benefits, including consistency, repeatability, and easier management of resources. Below is a guide on how to create AWS services using CloudFormation templates, along with the benefits of using this method.

Step1: Select a Region. For Example: Mumbai

Step2: Go to Cloud Formation in AWS

Step3: Save code using yaml/JSON formatted file

```
AWSTemplateFormatVersion: '2010-09-09'

Description: Create a VPC, two subnets, and an EC2 instance.


Resources:

  # Create VPC

  MyVPC:

    Type: AWS::EC2::VPC

    Properties:

      CidrBlock: 10.0.0.0/16

      EnableDnsSupport: true

      EnableDnsHostnames: true

      Tags:

        - Key: Name

          Value: MyVPC


  # Create Internet Gateway

  MyInternetGateway:

    Type: AWS::EC2::InternetGateway

  AttachGateway:

    Type: AWS::EC2::VPCGatewayAttachment

    Properties:
```

```
    VpcId: !Ref MyVPC
    InternetGatewayId: !Ref MyInternetGateway

# Create Public Route Table
PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref MyVPC

# Create Route to Internet
PublicRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref MyInternetGateway

# Subnet 1
Subnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref MyVPC
    CidrBlock: 10.0.1.0/24
    AvailabilityZone: !Select [0, !GetAZs '']

# Subnet 2
Subnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref MyVPC
    CidrBlock: 10.0.2.0/24
    AvailabilityZone: !Select [1, !GetAZs '']
```

```
# Associate Subnet 1 with Public Route Table
Subnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref Subnet1
    RouteTableId: !Ref PublicRouteTable

# Create Security Group
MySecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow SSH and HTTP
    VpcId: !Ref MyVPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0

# Launch EC2 Instance in Subnet 1
MyEC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t2.micro
    ImageId: ami-0e35ddab05955cf57  # Replace with a valid AMI ID for your region
    SubnetId: !Ref Subnet1
    SecurityGroupIds:
      - !Ref MySecurityGroup
    KeyName: my-key-pair2  # Replace with your key pair name
```

Step4: Create Stack.

Step5: Upload a template file, Upload your template directly to the console.(code should be in JSON or YAML formatted file).

Step6: Give stack name and submit (create it)

Outputs:

```
  VPCId:
    Description: VPC ID
    Value: !Ref MyVPC
  Subnet1Id:
    Description: Subnet 1 ID
    Value: !Ref Subnet1
  Subnet2Id:
    Description: Subnet 2 ID
    Value: !Ref Subnet2
  InstanceId:
Description: EC2 Instance ID
    Value: !Ref MyEC2Instance
```

Take a screenshot of EC2 instances/VPC/Subnet/Internet Gateway (any 3)

S3 Bucket

An **S3 bucket** is a storage container provided by **Amazon S3 (Simple Storage Service)**, part of Amazon Web Services (AWS). You can think of it as a folder in the cloud where you store and organize your data. Here are the key concepts:

◆ **Basics of an S3 Bucket:**

- **Name**: Globally unique across all AWS accounts.
- **Region**: Created in a specific AWS region.
- **Objects**: Files stored in a bucket. Each object has a key (its unique identifier), data, and metadata.
- **Folders**: Not real directories—just part of the object key path to organize data.

◆ **Common Use Cases:**

- Static website hosting
- Backup and restore
- Data archiving
- Media storage (images, videos, etc.)
- Big data analytics storage

◆ **Key Features:**

- **Versioning**: Keep multiple versions of an object.
- **Access control**: Managed via IAM policies, bucket policies, and ACLs.
- **Lifecycle policies**: Automate transition to cheaper storage classes or deletion.
- **Encryption**: Support for server-side and client-side encryption

Experiment 5:

How to create a object using S3 bucket and how the move the object to folder.

Step 1: Create a bucket. Bucket name should globally unique.

Step 2**:** Choose one file from the desktop.

Step 3: Disable the bucket public option.

Step 4: Enabled the bucket versioning and save it.

Step5: Upload one file from the desktop.

Step 6: Create a folder. Click on the file, go to action and move it.

Step 7: Click on browse option, Select the folder and choose the destination.

Take a screenshot of both File and Folder you created.

Experiment 6

How to Host a Static Website using S3 Bucket

Step1: Create a bucket.

Step2: Upload the folder.

Step3: disable the bucket public option.

Step4: enable static website option.

Step5: enable ACL permission.

Step6: make it public ACL for all the folder object.

Step7: Click on html folder and host it.

Step 8: Check whether it is working properly or not

Take a screenshot of uploading folder page, and static website which displayed on chrome(Any 3)

Cloud Watch

Amazon CloudWatch is a monitoring and observability service offered by AWS. It helps you collect and analyze operational data in the form of logs, metrics, and events, providing insights into your AWS infrastructure, applications, and services.

Key Features of Amazon CloudWatch

Metrics Monitoring

Tracks performance metrics for AWS services like EC2, RDS, Lambda, ECS, etc.

You can also publish custom metrics (e.g., application-specific data).

Logs

CloudWatch Logs lets you collect and store logs from:

AWS services (like Lambda or ECS)

Applications using agents (e.g., CloudWatch Agent or Fluent Bit)

Supports log filtering, searching, and insights queries.

Alarms

Set thresholds on metrics to trigger notifications via SNS or initiate auto-scaling actions.

For example, send an alert if CPU usage exceeds 80% for more than 5 minutes.

Experiment 7

How to create a alarm in Cloud Watch using a Simple Notifiaction Service.

Step 1: Create a SNS Topic and Subscribe your mail and mobile number.

Step 2: Create a Virtual Machine and Set your SNS Service Topics and Subscription.

Step 3: Create a Cloud Watch alarm with CPU UTILIZATION Metric.

Take a screenshot of email and text message(Confirmation)

Final output (In-Alarm or OK)