CLOUD COST ESTIMATOR

PROJECT

PREPARED BY:

B NANDINI TECHNICAL TRAINER

3BR23CS025 SIGNATURE

|  |
| --- |
| SI NO. CONTENTS |
| 1. AIM |
| 1. SYTEM REQIREMENTS |
| 1. ALGORITHM |
| 1. CODE |
| 1. EXPLANATION |
| 1. HOW CLOUD COST ESTIMATOR WORK |
| 1. COSTS KEY FACTORS AFFECTING CLOUD |
| 1. BENEFITS OF CLOUD COST ESTIMATORS |
| 1. POPULAR CLOUD COST ESTIMATORS |
| 1. CONCLUSION |
|  |
|  |

INDEX :

AIM:

* To create a Python-based application that accurately estimates the monthly cloud cost for a given cloud infrastructure setup on a chosen provider (AWS, Azure, GCP).
* inputting details about resource requirements, usage To provide a user-friendly interface for patterns, and service selections.
* To accurately estimate and compare cloud computing costs across multiple providers, enabling informed decision-making and optimized cloud infrastructure spending

SYSTEM REQUIREMENTS:

Hardware Requirements:

- 2 GHz dual-core processor

- 4 GB RAM (8 GB recommended)

- 100 GB disk space

- 64-bit Windows, Linux, or macOS

Software Requirements:

- Python, Java, or C# programming language

- Web framework (Flask, Django, Spring Boot)

- Database (MySQL, PostgreSQL, MongoDB)

- Front-end framework (React, Angular, Vue.js)

Functional Requirements:

- User authentication and authorization

- Cloud resource data input/validation

- Cost estimation algorithms

- Real-time cost updates/notifications

- Reporting/analytics for cost optimization

Non-Functional Requirements:

- Scalability

- Performance

- Security

- Usability

- Reliability

Cloud Provider Requirements:

- AWS IAM roles/API keys

- Azure subscription IDs/API keys

- Google Cloud project IDs/API keys

Additional Requirements:

- Documentation

- Testing (unit, integration, user acceptance)

- Maintenance (updates, bug fixes, security patches)

- User support/training/consulting services

ALGORITHM:

STEP 1: Define Classes

- CloudProvider: Stores cloud provider details (name, pricing).

- CostEstimate: Manages CRUD operations for cost estimates (create, read, update, delete).

- CostCalculator: Handles cost calculation and comparison algorithms.

STEP 2: CRUD Operations

- Create: Store a new cost estimate with a unique ID, provider, usage, and cost.

- Read: Retrieve an estimate by ID.

- Update: Modify existing estimate details using the ID.

- Delete: Remove an estimate by ID.

STEP 3: Cost Calculation

- Multiply resource usage (CPU, memory, etc.) by the provider’s pricing for each resource.

- Sum the calculated values to get the total estimated cost.

STEP 4: Cost Comparison

- For each provider, calculate the total cost based on the same usage details.

- Return a comparison of costs across providers.

CODE:

class CloudServiceProvider:

def \_init\_(self, name, cpu\_cost, memory\_cost, storage\_cost):

self.name = name

self.cpu\_cost = cpu\_cost # Cost per CPU per hour

self.memory\_cost = memory\_cost # Cost per GB of RAM per hour

self.storage\_cost = storage\_cost # Cost per GB of storage per month

def calculate\_cost(self, cpus, memory, storage, hours=1):

""" Calculate cost for the given resource usage """

return (self.cpu\_cost \* cpus \* hours) + (self.memory\_cost \* memory \* hours) + (self.storage\_cost \* storage)

def \_str\_(self):

return f"{self.name}: CPU Cost: ${self.cpu\_cost}/hour, Memory Cost: ${self.memory\_cost}/GB/hour, Storage Cost: ${self.storage\_cost}/GB/month"

class CloudCostEstimator:

def \_init\_(self):

self.providers = []

# CRUD Operations

def add\_provider(self, provider):

self.providers.append(provider)

print(f"{provider.name} has been added.")

def update\_provider(self, provider\_name, new\_cpu\_cost, new\_memory\_cost, new\_storage\_cost):

for provider in self.providers:

if provider.name == provider\_name:

provider.cpu\_cost = new\_cpu\_cost

provider.memory\_cost = new\_memory\_cost

provider.storage\_cost = new\_storage\_cost

print(f"{provider.name} has been updated.")

return

print(f"Provider {provider\_name} not found.")

def delete\_provider(self, provider\_name):

for provider in self.providers:

if provider.name == provider\_name:

self.providers.remove(provider)

print(f"{provider\_name} has been removed.")

return

print(f"Provider {provider\_name} not found.")

def list\_providers(self):

if not self.providers:

print("No providers available.")

for provider in self.providers:

print(provider)

# Cost Comparison

def compare\_costs(self, cpus, memory, storage, hours):

print(f"\nComparing costs for {cpus} CPUs, {memory}GB Memory, {storage}GB Storage for {hours} hours:\n")

for provider in self.providers:

cost = provider.calculate\_cost(cpus, memory, storage, hours)

print(f"{provider.name}: Total Cost = ${cost:.2f}")

# Function to get user input for provider details

def get\_provider\_input():

name = input("Enter the cloud provider name: ")

cpu\_cost = float(input(f"Enter the CPU cost per hour for {name}: "))

memory\_cost = float(input(f"Enter the memory cost per GB per hour for {name}: "))

storage\_cost = float(input(f"Enter the storage cost per GB per month for {name}: "))

return CloudServiceProvider(name, cpu\_cost, memory\_cost, storage\_cost)

# Main function for user interaction

def main():

estimator = CloudCostEstimator()

while True:

print("\n--- Cloud Cost Estimator ---")

print("1. Add Cloud Provider")

print("2. Update Cloud Provider")

print("3. Delete Cloud Provider")

print("4. List Cloud Providers")

print("5. Compare Costs")

print("6. Exit")

choice = input("Enter your choice: ")

if choice == "1":

provider = get\_provider\_input()

estimator.add\_provider(provider)

elif choice == "2":

name = input("Enter the cloud provider name to update: ")

cpu\_cost = float(input(f"Enter the new CPU cost per hour for {name}: "))

memory\_cost = float(input(f"Enter the new memory cost per GB per hour for {name}: "))

storage\_cost = float(input(f"Enter the new storage cost per GB per month for {name}: "))

estimator.update\_provider(name, cpu\_cost, memory\_cost, storage\_cost)

elif choice == "3":

name = input("Enter the cloud provider name to delete: ")

estimator.delete\_provider(name)

elif choice == "4":

estimator.list\_providers()

elif choice == "5":

cpus = int(input("Enter the number of CPUs: "))

memory = float(input("Enter the amount of memory (in GB): "))

storage = float(input("Enter the amount of storage (in GB): "))

hours = int(input("Enter the number of hours: "))

estimator.compare\_costs(cpus, memory, storage, hours)

elif choice == "6":

print("Exiting the program.")

break

else:

print("Invalid choice. Please try again.")

if \_name\_ == "\_main\_":

main()

SAMPLE OUTPUT:

--- Cloud Cost Estimator ---

1. Add Cloud --- Cloud Cost Estimator ---

1. Add Cloud Provider

2. Update Cloud Provider

3. Delete Cloud Provider

4. List Cloud Providers

5. Compare Costs

6. Exit

Enter your choice: 1

Enter the cloud provider name: AWS

Enter the CPU cost per hour for AWS: 600

Enter the memory cost per GB per hour for AWS: 500

Enter the storage cost per GB per month for AWS: 700

AWS has been added.

--- Cloud Cost Estimator ---

1. Add Cloud Provider

2. Update Cloud Provider

3. Delete Cloud Provider

4. List Cloud Providers

5. Compare Costs

6. Exit

Enter your choice: 2

Enter the cloud provider name to update: AWS

Enter the new CPU cost per hour for AWS: 300

Enter the new memory cost per GB per hour for AWS: 400

Enter the new storage cost per GB per month for AWS: 500

AWS has been updated.

--- Cloud Cost Estimator ---

1. Add Cloud Provider

2. Update Cloud Provider

3. Delete Cloud Provider

4. List Cloud Providers

5. Compare Costs

6. Exit

Enter your choice: 1

Enter the cloud provider name: EC6

Enter the CPU cost per hour for EC6: 400

Enter the memory cost per GB per hour for EC6: 500

Enter the storage cost per GB per month for EC6: 600

EC6 has been added.

Provider

2. Update Cloud Provider

3. Delete Cloud Provider

4. List Cloud Providers

5. Compare Costs

6. Exit

Enter your choice: 4

AWS: CPU Cost: $300.0/hour, Memory Cost: $400.0/GB/hour, Storage Cost: $500.0/GB/month

EC6: CPU Cost: $400.0/hour, Memory Cost: $500.0/GB/hour, Storage Cost: $600.0/GB/month

--- Cloud Cost Estimator ---

1. Add Cloud Provider

2. Update Cloud Provider

3. Delete Cloud Provider

4. List Cloud Providers

5. Compare Costs

6. Exit

Enter your choice: 5

Enter the number of CPUs: 2

Enter the amount of memory (in GB): 500

Enter the amount of storage (in GB): 6

Enter the number of hours: 6

Comparing costs for 2 CPUs, 500.0GB Memory, 6.0GB Storage for 6 hours:

AWS: Total Cost = $1206600.00

EC6: Total Cost = $1508400.00

--- Cloud Cost Estimator ---

1. Add Cloud Provider

2. Update Cloud Provider

3. Delete Cloud Provider

4. List Cloud Providers

5. Compare Costs

6. Exit

Enter your choice: 3

Enter the cloud provider name to delete: AWS

AWS has been removed.

--- Cloud Cost Estimator ---

1. Add Cloud Provider

2. Update Cloud Provider

3. Delete Cloud Provider

4. List Cloud Providers

5. Compare Costs

6. Exit

Enter your choice: 6

Exiting the program.

EXPLANATION:

A cloud cost estimator is a tool that helps you predict the potential costs of running your applications or workloads on a cloud platform. It's essential for businesses to have a clear understanding of their cloud spending to make informed decisions about their cloud infrastructure.

How Cloud Cost Estimators Work:

Resource Selection: You typically start by selecting the resources you plan to use, such as virtual machines (VMs), storage, databases, and network bandwidth.

Specification Input: For each resource, you provide details like instance type, storage capacity, and network bandwidth.

Usage Estimation: You estimate the expected usage of these resources, including factors like CPU utilization, memory consumption, and I/O operations.

Pricing Information: The estimator then applies the pricing information provided by the cloud provider to calculate the estimated costs.

Cost Breakdown: The tool often provides a detailed breakdown of costs, showing the contributions from different resources and usage patterns.

Costs Key Factors Affecting Cloud

Resource Type: Different resources have varying pricing structures. For example, on-demand instances might be more expensive than reserved instances.

Resource Size: The size of a resource (e.g., VM instance type) directly impacts its cost. Larger instances typically cost more.

Usage Patterns: How you use your resources affects costs. For instance, if you have consistent usage, reserved instances might be more cost-effective.

Region and Availability Zone: Costs can vary based on the geographic location of the resources.

Additional Services: Services like data transfer, object storage, and managed databases have their own pricing models.

Benefits of Cloud Cost Estimators

Planning Cost: Helps you estimate and budget for your cloud spending.

Optimized Resource Selection: Guides you in choosing the most cost-effective resources for your workload.

Cost Reduction: Identifies opportunities to reduce costs through resource optimization and usage adjustments.

Risk Mitigation: Helps you avoid unexpected cost overruns.

Popular Cloud Cost Estimators

Amazon AWS Cost Calculator: Offered by Amazon Web Services.

Google Cloud Cost Estimator: Provided by Google Cloud Platform.

Microsoft Azure Cost Calculator: Available from Microsoft Azure.

CONCLUSION:

The cloud cost estimator project successfully developed a tool that accurately predicts cloud infrastructure costs based on various input parameters. By providing users with reliable cost estimates, the tool empowers them to make informed decisions about their cloud deployments, optimize resource utilization, and effectively manage their cloud budgets.