ASTON TECHNOLOGIES

Table of Contents

Module 1: Basic Infrastructure

In this module we will be setting up a basic infrastructure to support a simple web application. Engineers should have a good understanding of basic infrastructure services available in AWS as well as how to configure them. This project is intended to be done with a newly created account to take advantage of certain free tier offerings. Any services or configurations done outside of the scope of this document may result in exceeding the limits of the free tier causing a charge to appear on the account.

Deliverables for this lab:

- **Module 1:**
    1. **Develop a diagram for Module 1 detailing all infrastructure components deployed.**
        - See **Module 2: Documentation** for more details.
- **Module 3:**
    1. **Lift & Shift for the current implementation (1st Diagram)**
    2. **Future implementation (2nd Diagram. Show the Improvements)**
    3. **Cost analysis of both implementations.**
    4. **Technical implementation of the current architecture (Demo)**
        - **Use only free tier resources; only need to show the services implemented.**

1.01 | Create an Account

1. Go to the AWS website: https://aws.amazon.com/
2. Click on the "Create an AWS Account" button.
3. Choose "Create a new AWS account."
4. Enter your email address and choose a password.
5. Enter your contact information, including your name, address, and phone number.
6. Enter your payment information. AWS requires a valid credit card to verify your identity and prevent fraud. Only free tier options will be used in this lab.
7. Choose the AWS free plan.
8. Review the terms and conditions and click "Create Account and Continue."

Once you have completed these steps, AWS will send you an email to confirm your account. Follow the instructions in the email to verify your email address and complete the account setup process. Once your account is verified, you can start using AWS services and tools.

1.02 | Create an IAM User

Identity and Access Management (IAM) is a service that enables you to manage users and their access to AWS resources. IAM provides features like creating and managing users, groups, and roles, setting permissions, and creating policies to define permissions. When you first create an AWS account, you begin with the root user (the email used when creating the account). It's strongly recommended that you don't use the root user for everyday tasks, even the administrative ones. Best practice is to only use the root user to perform a few account and service management tasks. A list of these tasks that can only be performed on the root user can be found here. Instead, IAM users should be created and given permissions based on the tasks they will be performing, following the principle of least privilege. We will create an IAM user with administrative permissions and utilize that user for the remainder of the lab.

1. In the search bar, type "IAM" and select the service.
2. In the left-hand navigation menu, click on "Users".
3. Click the "Add User" button at the top of the screen.
4. Enter a name for the new user in the "Username" field.
5. Select the "AWS Management Console access" checkbox to enable console access for the user.
6. Select "Programmatic access" if you want the user to be able to access AWS using the API or command line.
7. Check "I want to create an IAM user".
8. Type your password for the user and uncheck the box for creating a new password at next sign in.
9. Choose "Require password reset" if you want the user to reset their password the first time they log in.
10. Click "Next".
11. Select the "Add user to group" radio button.
12. Name the group "ADMIN-USERS'
13. Search for "AdministratorAccess" in the search bar.
14. Scroll down and click "Create User Group".
15. Select the group and click "Next".
16. Click "Create User"
17. Add any tags you want to assign to the user and click "Next: Review".
18. Review the user's details and permissions.
19. Click "Create user".

1.03 | Create an Instance Role

In AWS, an instance profile is a container for an AWS Identity and Access Management (IAM) role that you can use to pass role information to an EC2 instance when the instance starts. Instance profiles enable EC2 instances to access AWS services and resources securely without the need to store long-term credentials or access keys on the instance.

An instance profile consists of an IAM role, which defines a set of permissions that determine what an EC2 instance can and cannot do. When you launch an EC2 instance, you can specify the instance profile to associate with the instance, and the instance can then assume the permissions defined in the associated IAM role.

Instance profiles simplify the management of AWS access permissions for EC2 instances by allowing you to manage permissions at the role level, rather than at the instance level. This makes it easier to grant or revoke access to AWS services and resources as needed, without having to modify permissions for individual instances.

Instance profiles are particularly useful in environments where you need to launch multiple instances with the same permissions. By creating a single IAM role and associating it with an instance profile, you can launch as many instances as needed with the same permissions, without having to configure permissions for each instance individually.

1. Select “Roles” from the IAM side menu.
2. Click the "Create role" button at the top of the screen.
3. Select "AWS service" as the trusted entity and "EC2" as the service that will use this role.
4. Click "Next".
5. Type “S3” in the search bar. Select “AmazonS3FullAccess”.
6. Enter a name for the role in the "Role name" field.
7. Click "Create role".

1.04 | Create a VPC

A Virtual Private Cloud (VPC) is a logically isolated virtual network that you can create in your AWS account. You can think of a VPC as a virtual data center in the cloud that allows you to launch AWS resources, such as EC2 instances (Servers), RDS databases, and Elastic Load Balancers, in a virtual network that you control.

Here are some common use cases for VPCs in AWS:

- Security: VPCs allow you to define your own network topology and control inbound and outbound traffic to your resources. You can create network access control lists (ACLs) and security groups to filter traffic at the subnet and instance level and use VPN or Direct Connect to establish a secure connection between your on-premises data center and your VPC.
- Resource isolation: VPCs allow you to isolate your AWS resources from each other and from the public internet. You can create multiple subnets and place resources in different subnets based on their security requirements or application needs. You can also use VPC peering to connect VPCs within the same account or across different accounts, enabling resources to communicate with each other securely.
- Network customization: VPCs allow you to customize your network architecture by choosing your own IP address range, creating subnets, and configuring route tables and internet gateways.
- Compliance: VPCs can help you meet regulatory compliance requirements by allowing you to control network traffic and access to your resources.

In this lab we will be setting up a basic VPC to customize our network and launch our servers in.

1. Go to the Amazon VPC console by selecting "Services" and searching for "VPC."
2. Click on the "Create VPC" button.
3. Ensure that “VPC only” is selected during the creation of the VPC.
4. In the "Create VPC" wizard, enter a name for your VPC – “My VPC 1.”
5. In the "IPv4 CIDR block" field, enter "10.0.0.0/16."
6. Select the appropriate "Tenancy" option; "Default" is what we will be using.
7. Click the "Create" button to create your VPC.

1.05 | Create a Public Subnet

A public subnet is a subnet that has a direct route to the internet via an internet gateway (IGW). Instances launched in a public subnet can have a public IP address and can be accessed from the internet. Public subnets are typically used for resources that need to be publicly accessible, such as web servers, load balancers, or other publicly facing resources. Resources in a public subnet can communicate with the internet, but they can also communicate with other resources in private subnets within the same virtual private cloud (VPC) using their private IP addresses.

1. In the VPC dashboard, select "Subnets" from the left-hand menu.
2. Click the "Create subnet" button located at the top of the screen.
3. In the "Create subnet" window, select your VPC named “My VPC 1.”
4. For the "Availability Zone," select “US-EAST-1.
5. For the "IPv4 CIDR block," enter "10.0.1.0/24".

6. Leave the "Auto-assign IP" checkbox selected.
7. For the "Name tag," enter the name “My Public Subnet 1”.
8. Click the "Create" button.
9. Select your subnet -> Actions -> Edit subnet settings.
10. Enable auto-assign public IPv4 address.
11. Click the “Save” button.
12. Go through the same steps with a second subnet named “My Public Subnet 2” and a CIDR of 10.0.2.0/24. Ensure the availability zone is not the same between the two.

1.06 | Create an Internet Gateway

An Internet Gateway (IGW) serves as the entry and exit point of traffic for your VPC. It allows traffic to flow between your VPC and the internet, enabling internet access for resources within the VPC. After you create a VPC you then attach the internet gateway to the VPC. Once attached, you can configure routing tables to allow or block traffic from the internet to your VPC resources.

1. In the VPC dashboard, select "Internet Gateways" from the left-hand menu.
2. Click the "Create internet gateway" button located at the top of the screen.
3. In the "Create internet gateway" window, give your Internet Gateway a name that's descriptive, such as "My Internet Gateway 1".
4. Click the "Create" button to create the Internet Gateway.
5. Once the Internet Gateway is created, select it in the list of Internet Gateways.
6. Click the "Actions" dropdown menu and select "Attach to VPC".
7. In the "Attach to VPC" window, select "My VPC 1" from the dropdown menu.
8. Click the "Attach" button.

1.07 | Create a Public Route Table

A public route table is a routing table that contains rules for routing traffic from your VPC's public subnets to the internet via the internet gateway. To enable access to the internet from your VPC, you typically associate a public route table with your VPC's public subnets. This allows instances within your public subnets to communicate with the internet through the internet gateway. A public route table is going to have a default route targeting the IGW, as well as other routes pointing to specific public subnets. It’s important to note that a public route table should not be associated with private subnets, as this can create security risks by allowing direct access to resources within the private subnet from the internet.

1. In the VPC dashboard, select "Route Tables" from the left-hand menu.
2. Click the "Create route table" button located at the top of the screen.
3. In the "Create route table" window, give your route table the name "Public Route Table".
4. Select the VPC we created earlier to associate with the route table.
5. Click the "Create" button to create the route table.
6. Once the route table is created, select it in the list of route tables.
7. Click the "Routes" tab at the bottom of the screen.
8. Click the "Edit routes" button.
9. In the "Edit routes" window, click the "Add route" button.
10. In the "Add route" window, set the "Destination" field to "0.0.0.0/0".
11. In the "Target" field, select "Internet Gateway" from the dropdown menu.
12. Select the Internet Gateway you previously created from the dropdown menu.
13. Click the "Save routes" button to save the new route.
14. Click the "Subnet associations" tab at the bottom of the screen.
15. Click the "Edit subnet associations" button.
16. In the "Edit subnet associations" window, select "My Public Subnet 1" from the list of available subnets. Do the same for the second subnet.
17. Click the "Save" button to associate the subnet with the route table.

1.08 | Create a Security Group

A security group is a virtual firewall that controls inbound and outbound traffic for EC2 instances or other resources within a VPC. You can use them to define rules to allow or restrict access based on specific protocols, ports, and source or destination IP addresses. Security groups are stateful, meaning that they keep track of the traffic that has been authorized, and they can automatically allow the response traffic to flow back to the originating instance.

1. In the VPC dashboard, select "Security Groups" from the left-hand menu.
2. Click the "Create security group" button located at the top of the screen.
3. In the "Create security group" window, give your security group the name "Web Server".
4. Add a description for the security group like “Public Webserver Security Group”.
5. Select "My VPC 1" as the VPC for the security group.
6. In the "Inbound rules" section, click the "Add rule" button.
7. In the "Add rule" window, set the "Type" field to "HTTP".
8. In the "Source" field, select "Anywhere – Ipv4" from the dropdown menu.
9. In the "Description" field, enter "Allow HTTP traffic".
10. Create another rule allowing SSH from your IP.
11. Click the "Create security group" button to create the security group.

1.09 | Create an EC2 Instance

An Amazon Elastic Compute Cloud (EC2) instance is a virtual server in AWS that provides computing capacity in the cloud. It allows you to create and configure virtual machines (VMs) on demand, which can be used for a variety of purposes such as hosting web applications, running databases, performing data analytics, or any other workload that requires computing resources.

They are highly configurable and can be customized to meet specific performance, security, and storage requirements. There are a number of ways to pay for instances with on-demand, reserved, spot, and dedicated host pricing models that will change the availability and price of the instance.

EC2 instances are available in a variety of sizes, or instance types, each optimized for different types of workloads. For example, some instance types are optimized for compute-intensive workloads, while others are optimized for memory-intensive workloads. These types will be chosen when setting up the instance. You can find a list of the different pricing models here.

When you launch an EC2 instance, you can choose the instance type, operating system, storage, and other configuration options. Once the instance is launched, you can connect to it remotely using a secure shell (SSH) or remote desktop protocol (RDP), depending on the operating system. You can also manage and monitor your EC2 instances using AWS tools such as AWS Management Console, AWS CLI, or third-party tools.

In this lab we will be configuring a basic EC2 instance, installing a web server, and utilizing launch templates with auto scaling for high availability and elasticity.

1. Navigate to the Amazon EC2 service by searching for "EC2" in the search bar or selecting it from the list of services.
2. Once in the EC2 dashboard, click the "Launch instance" button.
3. In the “Name and tags” section, use the name “My Web Server 1”.
4. In the "Application and OS Images (Amazon Machine Image) section, select “Amazon Linux 2023 AMI”. Ensure it says “Free tier eligible”.
5. In the "Choose an Instance Type" section, select "t2.micro" instance type. Ensure it says “Free tier eligible”.
6. In the “Key pair (login)” section, select “Proceed without a key pair”.
7. In the “Network Settings” section, select “Edit”.

8. In the "Configure Security Group" screen, select the "Select an existing security group" option.
9. Select the VPC you created, “My VPC 1” and the subnet “My Public Subnet 1”.
10. From the dropdown menu, select the "Web Server" security group that you previously created.
11. In the "Add Storage" screen, leave the storage settings at their default values.
12. Expand the “Advanced details” section and scroll down to the “User data – optional” portion.
13. Enter the following script:

```
#!/bin/bash
yum install httpd -y
yum update -y
service httpd start
chkconfig httpd on
echo "<html><h1>Hello World</h1></html>" >
/var/www/html/index.html
chmod 644 /var/www/html/index.html
```

14. Click the "Launch" button to launch your instance.
15. After a few minutes, copy the public IP of the EC2 instance the paste it in a web browser. You should see “Hello World” on the website.

1.10 | Create an S3 Bucket

Amazon S3 (Simple Storage Service) is a highly scalable and secure object storage service provided by AWS. An S3 bucket is a container for storing objects (files) within Amazon S3. It's similar to a folder in a file system, but with added features such as scalability, redundancy, and security.

An S3 bucket can be used to store and retrieve any type of data, including static files, images, videos, documents, and databases. The data stored in an S3 bucket can be accessed from anywhere in the world using simple HTTP or HTTPS protocols.

S3 buckets are highly scalable and can store virtually unlimited amounts of data. They are also highly available, durable, and secure. We can control access to their S3 buckets through permissions, policies, and encryption.

S3 buckets are used for a wide range of purposes, such as backup and archiving, data analytics, content distribution, and hosting static websites. In this lab we will store a script that our EC2 instances will download and execute to install the web services they'll be running.

1. Search for "s3" in the search bar.
2. Click the "Create bucket" button.
3. In the "Bucket name" field, enter a unique name.
4. Choose a region for the bucket. Use "us-east-1"
5. Leave all other options at their default values.
6. Click the "Create bucket" button at the bottom of the page.

1.11 | Upload to S3

1. On your computer, open a notepad file and paste the following text into it:

```
#!/bin/bash
sudo su
sudo yum install httpd -y
sudo yum update -y
sudo service httpd start
sudo chkconfig httpd on
sudo echo "Hello World from $(hostname -f)" >
/var/www/html/index.html/var/www/html/index.html
sudo chmod 644 /var/www/html/index.html
```

2. Save the file as awslabscript.sh.
3. Navigate to the s3 bucket you created.
4. Upload this file by clicking "Upload" and then "Add files".
5. Once uploaded, click on the file and copy the S3 URL. It should look similar to this "s3://myawsbucket3333/awslabscript.sh". We will use this link later.

1.12 | Launch Templates

A Launch Template is a feature that allows us to create pre-configured templates for launching instances on EC2. You can use these to quickly launch instances with consistent configurations, including instance type, AMI (Amazon Machine Image), security groups, key pairs, and other settings.

We can define and save configurations for instances, then use those templates to launch instances individually or in groups, with automatic scaling. This simplifies the process of launching instances so we don't need to manually configure each instance every time we want to launch one.

Launch Templates can also be used in conjunction with Auto Scaling groups to ensure that newly launched instances automatically conform to the desired configuration. We can utilize these to ensure that all instances are configured in the same way, which can improve security, reliability, and consistency. This lab will utilize launch templates in conjunction with Auto Scaling groups for this reason.

1. In the EC2 dashboard, click on "Launch Templates" on the left-hand menu and then click on the "Create Launch Template" button.
2. Give your launch template the name "webserver-template", and a description.
3. Under the "Choose an Amazon Machine Image (AMI)" section, select the appropriate AMI for your web server. Select the same AMI as the previous, manually configured EC2 instance: “Amazon Linux 2023 AMI”.
4. In the "Instance Type" section, select t2.micro.
5. In the "Key Pair" section, keep the default “Don’t include in launch template”.
6. Under the "Security Groups" section, select the “Web Server” security group that was created earlier.
7. In the “Resource tags” section, add a Name with the value of “Web App Server” and set the “Resource types” to “Instances”.
8. In the “Advanced Details” section, under “IAM instance profile” select the instance role you created earlier.
9. In the “Advanced Details” section, under "User Data", add the following script:

```
#!/bin/bash
aws s3 cp <INSERT S3 URL HERE> /home/ec2-user/
chmod +x /home/ec2-user/awslabscript.sh
bash /home/ec2-user/awslabscript.sh
```

10. Under the "Advanced Details" section, ensure that no subnet is included in the template.

11. Review your launch template settings and click on the "Create Launch Template" button to create your launch template.

1.13 | Auto Scaling Group

An Auto Scaling group is a feature that automatically scales the number of EC2 instances in a group based on user-defined rules. The goal of an Auto Scaling group is to automatically adjust the number of EC2 instances in response to changes in demand, ensuring that the application or service hosted on the instances has the capacity to handle incoming traffic.

We can create an Auto Scaling group and specify the minimum and maximum number of instances to be maintained at any given time. When the demand for the application or service increases, the Auto Scaling group will automatically launch new instances to handle the traffic. When the demand decreases, the Auto Scaling group will terminate instances to save costs.

Auto Scaling groups work in conjunction with the Launch Templates we created earlier to define the type and configuration of the EC2 instances launched by the group.

We can define scaling policies based on metrics such as CPU utilization, network traffic, or custom metrics. These policies can be used to scale up or down the number of instances in the group and can be triggered manually or automatically. Overall, Auto Scaling groups are a powerful tool for managing the capacity of EC2 instances and ensuring that applications and services have the resources they need to handle traffic and are a key component for elasticity in the Cloud.

1. On the EC2 Dashboard, click on "Auto Scaling Groups" on the left-hand menu.
2. Create a name under the “Auto Scaling group name”.
3. Click on "Create Auto Scaling Group" and choose the launch template that was created previously named “webserver-template”. Click “Next”.
4. On the “Choose instance launch options” page, select your VPC and your two availability zones. Click “Next”.
5. On the “Configure Advanced Options” page, leave all values default. Click “Next”.
6. On the “Configure group size and scaling policies”, under "Group Size," set the minimum and maximum number of instances you want in the group. Set the “Desired capacity” to 2. Set the “Minimum capacity” to 2. Set the “Maximum capacity” to 3. Leave the other options at their default values. Click “Skip to Review”.
7. Click "Create Auto Scaling Group" to finish.

1.14 | Target Group

A target group is a logical container for routing requests to registered targets using AWS' load balancer. A set of resources, like EC2 instances or IP addresses, are set as the targets and traffic is routed to them based on the rules you define.

1. On the EC2 page, select the "Target Groups" option from the left-hand menu.
2. Click the "Create target group" button.
3. Select the "Instance" option for the "Target type" field.
4. Enter a name for your target group in the "Name" field called "Webserver-TG".
5. Choose the appropriate protocol and port for your web server instances from the "Protocol" and "Port" drop-down menus. Use HTTP port 80.
6. Under "VPC" select the VPC created earlier named "My VPC 1".
7. Under "Health Checks" leave the default values.
8. Click the "Next" button to proceed to the "Register targets" page.
9. Select the instances that you want to include in your target group from the list of available instances.
10. Click the "Add to registered" button to add the selected instances to the target group.
11. Click the "Create" button to create your target group and register the selected instances.

1.15 | Elastic Load Balancer

Elastic Load Balancing (ELB) is a service that automatically distributes incoming traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses, in one or more Availability Zones. This helps to improve the availability and fault tolerance of your applications by distributing traffic across multiple resources and can also help to improve the scalability of your applications by automatically scaling resources up or down based on traffic patterns.

There are four types of ELBs available in AWS:

- Application Load Balancer (ALB): It's designed to route incoming traffic to specific targets based on the content of the request. This allows you to route traffic to different services based on URL paths or hostnames. (L7).
- Network Load Balancer (NLB): It's designed to handle large volumes of traffic and forward it directly to the targets without modifying the request. This makes it ideal for handling TCP, UDP, and TLS traffic. (L4)
- Gateway Load Balancer (GWLB): It's design to allow you to load balance traffic to virtual network appliances such as firewalls, intrusion detection and prevention systems, and deep packet inspection systems. It is a regional service that operates within a single AWS region. Handles mainly IP traffic. (L3-4).

- Classic Load Balancer (CLB): It's the oldest of the four ELBs and is designed to handle HTTP, HTTPS, and TCP traffic. It provides basic load balancing features and is suitable for simple applications. However, it has been replaced by ALB and NLB for most use cases. (L3-4)

We will be creating an ALB for this lab.

1. In the left-hand menu, click on "Load Balancers."
2. Click on the "Create Load Balancer" button.
3. Select "Application Load Balancer" as the load balancer type.
4. Give your load balancer the name "Web Server LB" and set the scheme as internet-facing.
5. Under Network mapping, select your VPC and map to the only AZ in use with the appropriate public subnet.
6. Under Security groups, select the "Web Server" security group.
7. Under "Listeners," add a listener for HTTP traffic on port 80. Set the default action to forward to the "Webserver-TG" target group.
8. Click on the "Create" button to create your load balancer.
9. Once your load balancer is created, you will see a DNS name for your load balancer. Copy this name and paste it in your browser. You should see one of your web servers pull up. Hit refresh a few times to see it switching between the different servers.

1.16 | Clean-up

1. Search for "EC2" in the search bar. Select the service.
2. Click on "Load Balancers". Select your load balancer, select "Actions" then click "Delete load balancer".
3. Click on "Target Groups". Select the target group, select "Actions" then "Delete".
4. Click on "Auto Scaling Groups". Select the scaling group then click "Delete".
5. Click on "Instances". Select all EC2 instances, click "Instance State", then click "Terminate".
6. Search for "VPC" in the search bar. Select the service.
7. Click on "Your VPCs". Select your VPC, click "Actions", then click "Delete VPC".
8. Search for "S3" in the search bar. Select the service.
9. Enter into your S3 bucket and delete any files. Select the bucket and click "Delete".

Module 2: Documentation

In this module we will focus more on the documentation aspect of the process. A good solution is held together overtime with good documentation. There are several diagramming tools that can be used for AWS architecture design, each with its own strengths and weaknesses. Here are a few of the most popular and effective diagramming tools for AWS:

1. Lucidchart
2. Visio
3. Draw.io
4. Cloudcraft

There are a number of best practices for Cloud diagrams. AWS has a good breakdown of them listed in their PowerPoint toolkit on the architecture design page.

2.01 | Deliverable: Lab Diagram

Create a diagram for the infrastructure you deployed in Module 1. This deliverable must show the services being used and their relationship with each other. Additional requirements:

- Use official AWS Architecture Icons. Use the newest available.
- Ensure the readability and accurate depiction of Module 1 infrastructure.
- Be able to speak about the general traffic flow through the network.

Module 3: Business Application Migration

In this module you will analyze a business and design an AWS architecture solution. Create a full solution given the business requirements. You do not need to provide a working AWS implementation; utilization of free tier offerings for proof of concept is encouraged, but not required.

3.01 | Company Background & Problem Statement

- Company name: Sea Ice Creamery (SIC)
- Description: A small, California-based, Ice cream company that focuses on making ice cream exclusively for the Fishing industry.

Due to their innovative product, Sea Ice Creamery has gained an exponential growth in clientele. Investors forecast that the growth of this company will continue for the next few years. Though manual configuration and creation of resources worked well for the company in the past, they need to keep up with the demands of their rapidly growing clientele. Their on-premises architecture has constantly created performance and reliability issues. Frequent over-provisioning of their architecture has made it difficult for SIC to handle growth. Upgrades to the on-premises architecture, over the past year, have been inconsistent and time-consuming. As a result, Sea Ice Creamery wants to start taking advantage of AWS cloud services. Sea Ice Creamery feels that a switch to the cloud would be more cost-effective, scalable, secure, and reliable than their current on-premises solution.

The company needs a way to consistently deploy, manage, and update factory resources across multiple AWS services. The company has also expanded their proprietary ice cream making software to several locations in multiple countries and must start automating to keep growing. They would also like a way to consistently manage resources across multiple environments. They currently find it challenging to replicate their deployments to these countries. Given the proprietary ice cream making software involved, security, policies, and compliance procedures are taken very seriously.

3.02 | Project Requirements

As long as it is cost effective to move to the cloud, Sea Ice Creamery would like to move forward with transitioning their entire on-premises infrastructure to AWS. They would like a detailed diagram showing the architectural design of the newly migrated AWS infrastructure. They would also like technical implementation of the newly migrated infrastructure in AWS.

Sea Ice Creamery would like to know what it will cost them to manage their infrastructure per customer. They would like a comparison between on-premises costs VS costs in the cloud. Overall, the company would like to know how much it will cost in order to meet all of the required AWS technologies/services for successful implementation. Analysis (using some forms of AWS pricing tools) within a report-based document, displaying monthly/yearly and service usage costs, would best visualize the costs for SIC.

Though all detailed specs are not provided, it is up to you to finalize the pricing analysis with the most cost effective and performance efficient combination of AWS services/data requirements. The specifications you decide upon, using AWS cost analysis tools, should align with the migrated AWS architecture and be appropriate for Sea Ice Creamery's use-case. Backup and justify your calculated pricing in a report-based document. To get you and your team started on cost analysis, Sea Ice Creamery provided some brief data usage requirements when migrating to AWS:

- A minimum of two launched instances that must be as cost effective and performance efficient as possible. Can be used for up to 20 hours per day.
- Web servers will utilize a minimum of 1 vCPU and 1GB RAM.
- Database will utilize a minimum of 1 Core, 1vCPU, 2GB RAM, and a 3.3 GHz Processor.
- Architecture should be able to balance/handle up to 100 requests per second and be able to connect for 60 seconds.
- Highly available and scalable DNS web service that can handle up to 10 million standard queries per month.
- A minimum of one relational database service that is capable of up to 30 GB of data transferred out per month and 5 GB of data transferred in.
- An AWS Support service that can work best for Sea Ice Creamery.
- Future System Requirements: the items in this section only require report-based documentation; technical implementation not required.

In addition to the **architectural design, technical implementation, report and cost analysis**, Sea Ice Creamery would like to **design and document a future system**. The company would like these additions added to the architectural design (and possibly cost analysis), explained in a report document and diagram. They are unsure about what additional AWS resources to implement, but have created some requests:

- A highly available and scalable AWS architecture design, allowing SIC to manage resources across multiple environments.
- Ability to monitor the health of their resources at any point in time.

- Implementation of security for all resources deployed into the cloud and at each architectural layer.
- Use of an AWS tool to improve security and performance, reduce overall costs, and monitor service limits.
- Implementation of AWS automating solutions in order to keep growing.
- Ability to automatically adjust capacity to maintain steady, predictable performance.
- Implementation of a fast and flexible database solution.
- Ability to implement an object-based storage solution to retrieve data from anywhere.
- Expanding their current architectural layers/tiers by handling more than 5x the current workload and processing business logic.

As AWS consultants, they would like to make use of any additional cloud native tools that you think would help the company. The more strategic and combined use of AWS tools, the better (documentation only; no technical implementation required).

Deliverables:

- **Lift & Shift for the current implementation (Diagram)**
- **Future implementation (Diagram. Improvements)**
- **Cost analysis of both implementations.**
- **Technical implementation of the current architecture (Demo)**
    - **Use only free tier resources; only need to show the services implemented.**