Section 1: Multiple Choice Questions (MCQs)

1. c
2. a
3. c
4. a
5. b
6. b
7. a
8. b
9. b
10. c
11. a

Research-based AWS Questions - using google only:

12. A landing zone is a multi-account, well-architected AWS environment that serves as a launchpad for deploying workloads and applications. It offers a starting point for identity and access management, network design, data security, governance, multi-account architecture, and logging.
13. AWS WAF is a web application firewall that helps defend against attacks by letting you set up rules that allow, block, or track (count) web requests according to predetermined criteria.
    You may manage how your protected resources react to HTTP(S) web requests by using AWS WAF. Creating a web access control list (ACL) and linking it to one or more web application resources that you wish to safeguard is how you accomplish this. Incoming requests are forwarded to AWS WAF by the related resources so that the web ACL can review them. You can establish traffic patterns to search for in requests and what to do when a request matches by creating rules in your web ACL. Among the options for action are the following: Permit requests to be processed and answered by the protected resource. Block the requests. Count the requests. Run CAPTCHA or challenge checks against requests to verify human users and standard browser use.
14. Up to 1 petabyte of storage that is shaped like a suitcase and is used to manually move large data files that could be too heavy to send over the internet.
    Snowball is a petabyte-scale data transport service that moves massive volumes of data into and out of the AWS cloud using secure equipment.
    At speeds faster than the internet, AWS Snowball employs physical storage devices to move massive volumes of data between your on-site data storage location and Amazon Simple Storage Service (Amazon S3).
15. The methods used for manual snapshots and AWS Backup are extremely different. Multiple AWS services can be automatically backed up with AWS Backup, a centralized solution. While manual resource snapshots need the maintenance of each resource independently and lack the automation and widespread protection that AWS Backup provides, AWS Backup offers sophisticated tools for cross-region copying, comprehensive policy administration, and unified retention methods. AWS Backup offers more robust, scalable, and effective solutions that offer increased security, customizable

scheduling, and multi-service support, even though manual snapshots can be used for point-in-time backup.

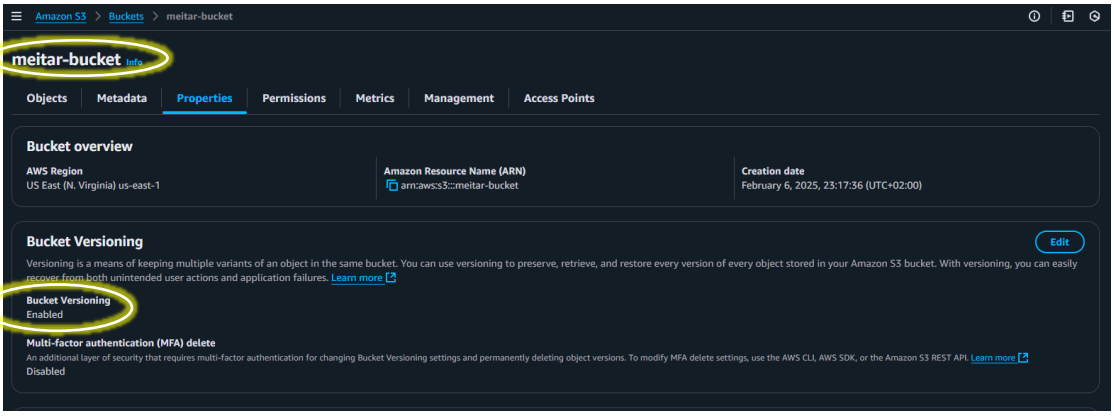
16. To detect malicious traffic, AWS Shield continuously monitors and examines traffic using anomaly algorithms, traffic signatures, and other types of analysis. Automatic resource scaling and inline deployments that filter harmful traffic are implemented in the event that an attack is identified, allowing legitimate requests to flow through. For transport layer attacks, the majority of this occurs in less than a second.
17. VPC Peering establishes direct one-to-one connections between VPCs, making it perfect for smaller configurations ,while Transit Gateway serves as a central gateway for linking several VPCs and on-premises networks at scale. While VPC Peering offers lower latency and simpler routing, Transit Gateway offers more sophisticated capabilities including dynamic routing.
18. With AWS Step Functions, which is a serverless management solution that aids in workflow automation, developers can create and oversee intricate, multi-step processes in the cloud. It provides a visual interface for creating workflows as state machines and connects several AWS services. Step Functions simplifies the development of distributed systems by automating decision-making, parallel processing, and error-handling. This reduces the complexity of application logic and improves system reliability. This implies that you can modularize your code into "Steps" and let AWS Step Functions handle the specifics rather than addressing partial failure cases, retries, and other "non happy cases" as part of your application logic.
19. AWS Control Tower is a service that offers a centralized, automated method of account governance and compliance, making it easier to manage several AWS accounts. It provides automated account provisioning, built-in best practices, centralized billing, customizable controls, enhanced security and compliance with predefined guardrails, centralized governance with a single dashboard for monitoring and managing all accounts, and quick multi-account setup in less than 30 minutes. In order to ensure flexible governance, Control Tower also makes account creation and configuration easier, lowers manual overhead, and enables businesses to apply and enforce particular policies across numerous accounts or units.
20. AWS Outposts, which bring AWS infrastructure to on-premises settings, is a crucial part of hybrid cloud solutions. It enables businesses to run AWS services locally while keeping cloud operations consistent. Low latency performance for real-time processing, data residency compliance for sectors with stringent requirements, and a consistent hybrid experience are some of the main advantages. Outposts is perfect for applications like modernization, edge computing, industrial automation, and retail analytics because it supports AWS services including EC2, EBS, S3, and RDS. It helps companies to build scalable, adaptable hybrid infrastructures that fulfill performance and compliance standards while increasing productivity and cutting expenses.
21. EFS: Shared file storage for several EC2 instances, perfect for programs requiring an elastic, scalable file system.
    S3: Cost-effective storage classes and global access for large-scale, unstructured data, such as media files and backups .
    EBS: Block storage for EC2 instances, which offers persistent, low-latency

storage and is ideal for applications like databases that are performance-sensitive.

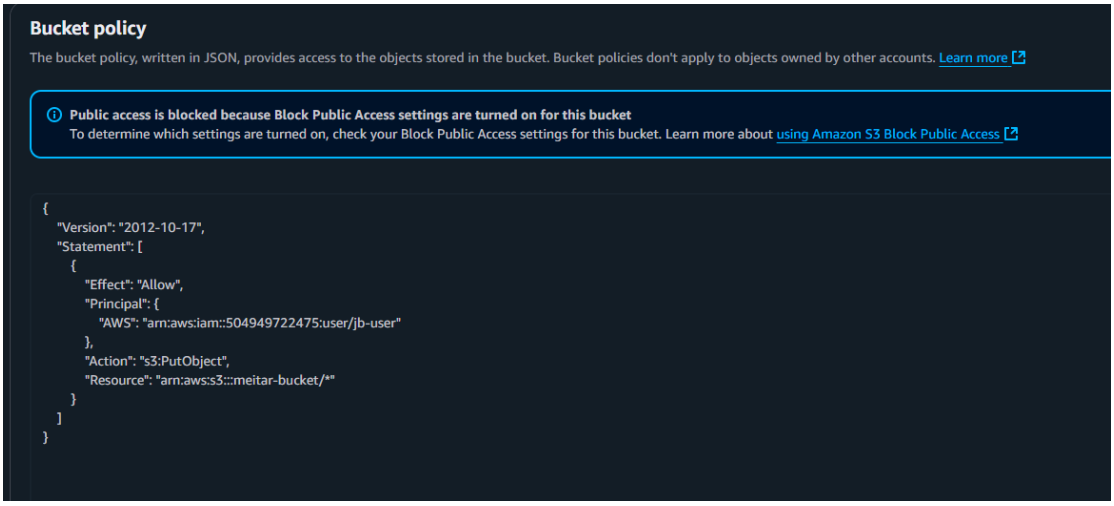
Section 2: Hands-on UI-Based Questions

1. S3 Bucket Configuration:

   created a bucket named "meitar-bucket", and enabled versioning



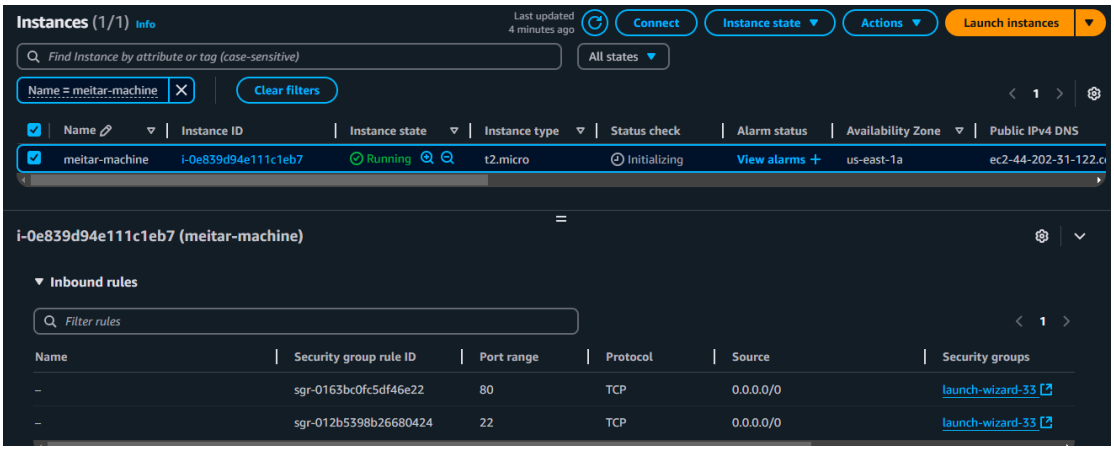
   also added policy for allowing only IAM user to upload objects:



2. **Launch an EC2 Instance**

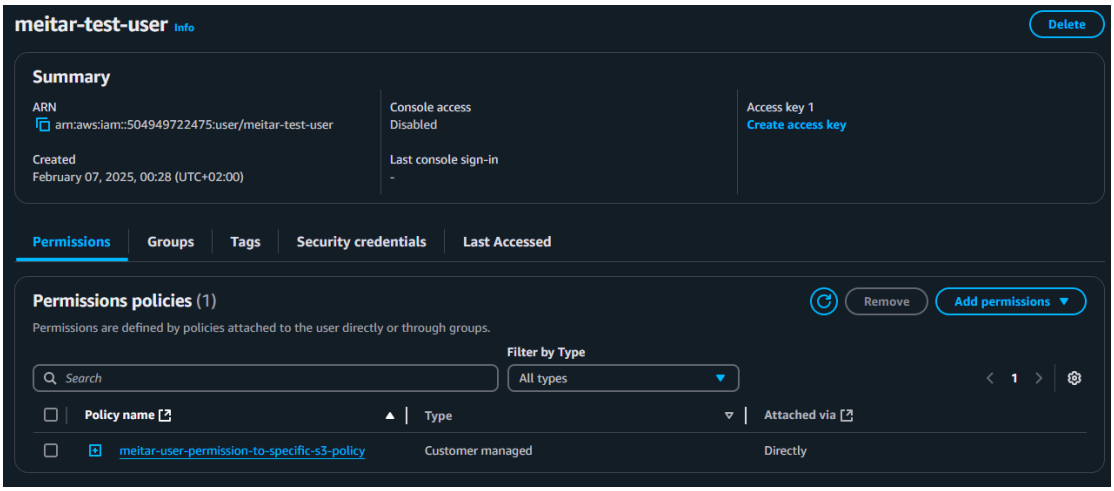
   created an EC2 instance called "meitar-machine" and configured inbound rules, as requested:

   the public ip address assigned to it: 44.202.31.122

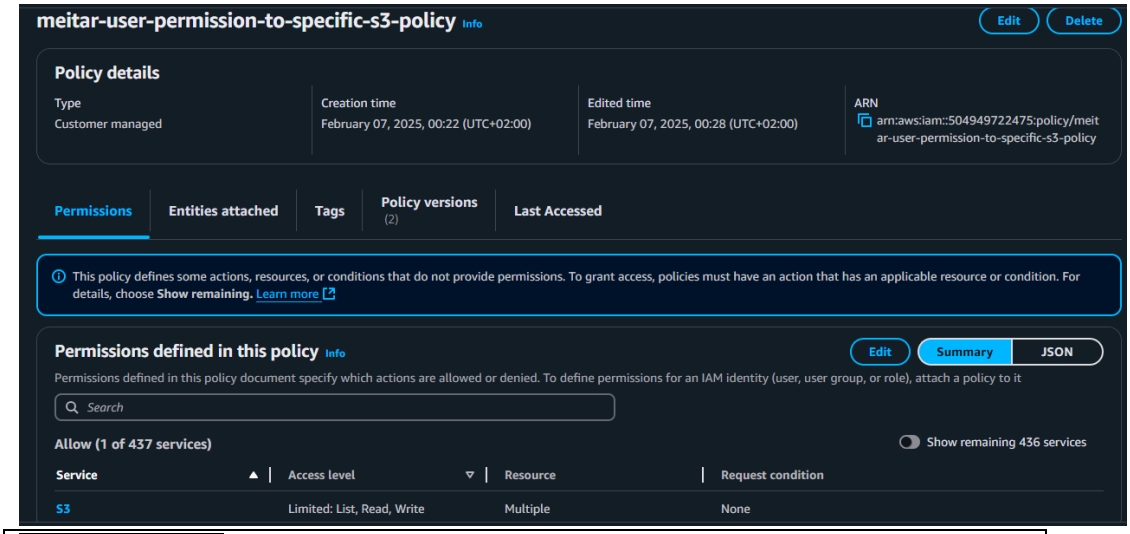


3. Configure an IAM User with S3 Access

created AMI user with the requested permission:



the policy permissions:

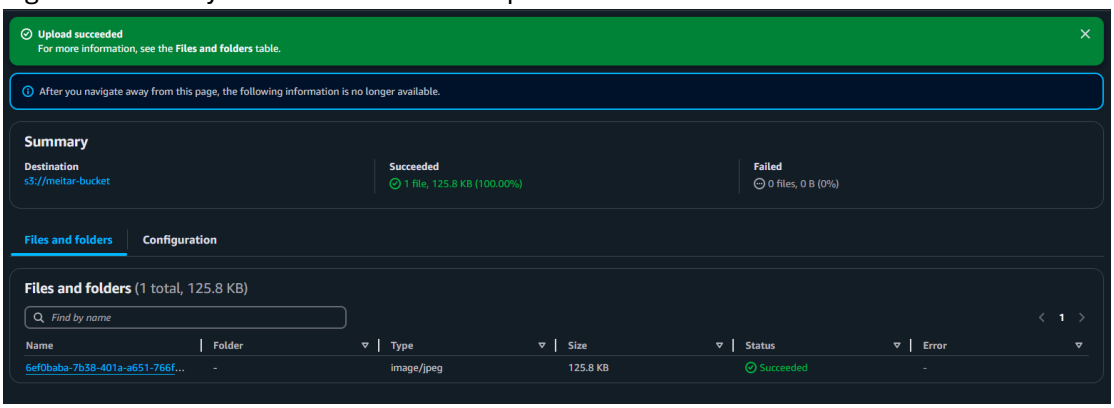


the JSON script:

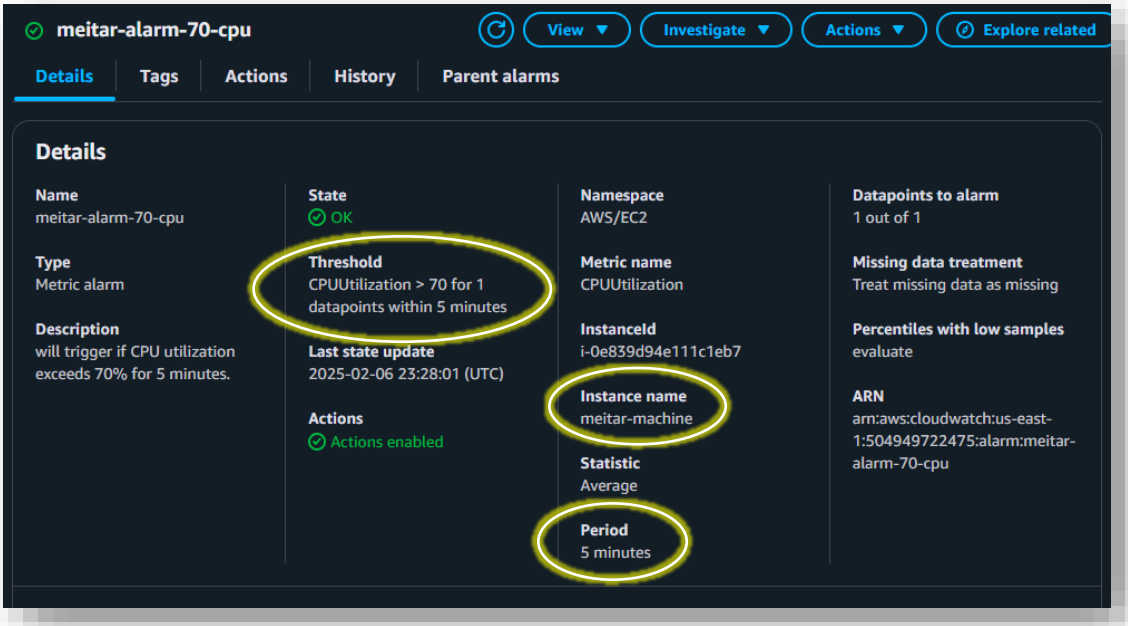
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::meitar-bucket/*",
        "arn:aws:s3:::meitar-bucket"
      ]
    },
    {
      "Sid": "VisualEditor2",
      "Effect": "Allow",
      "Action": [
        "s3:ListAccessPointsForObjectLambda",
```

```
            "s3:ListBucketMultipartUploads",
            "s3:GetBucketLocation",
            "s3:ListAccessPoints",
            "s3:ListCallerAccessGrants",
            "s3:ListBucketVersions",
            "s3:ListJobs",
            "s3:ListBucket",
            "s3:ListMultiRegionAccessPoints",
            "s3:ListStorageLensGroups",
            "s3:ListAccessGrantsLocations",
            "s3:ListMultipartUploadParts",
            "s3:ListStorageLensConfigurations",
            "s3:ListTagsForResource",
            "s3:ListAllMyBuckets",
            "s3:ListAccessGrantsInstances",
            "s3:ListAccessGrants"
        ],
        "Resource": "arn:aws:s3:::*"
    }
  ]
}
```

signed in with my user and was able to upload a file:

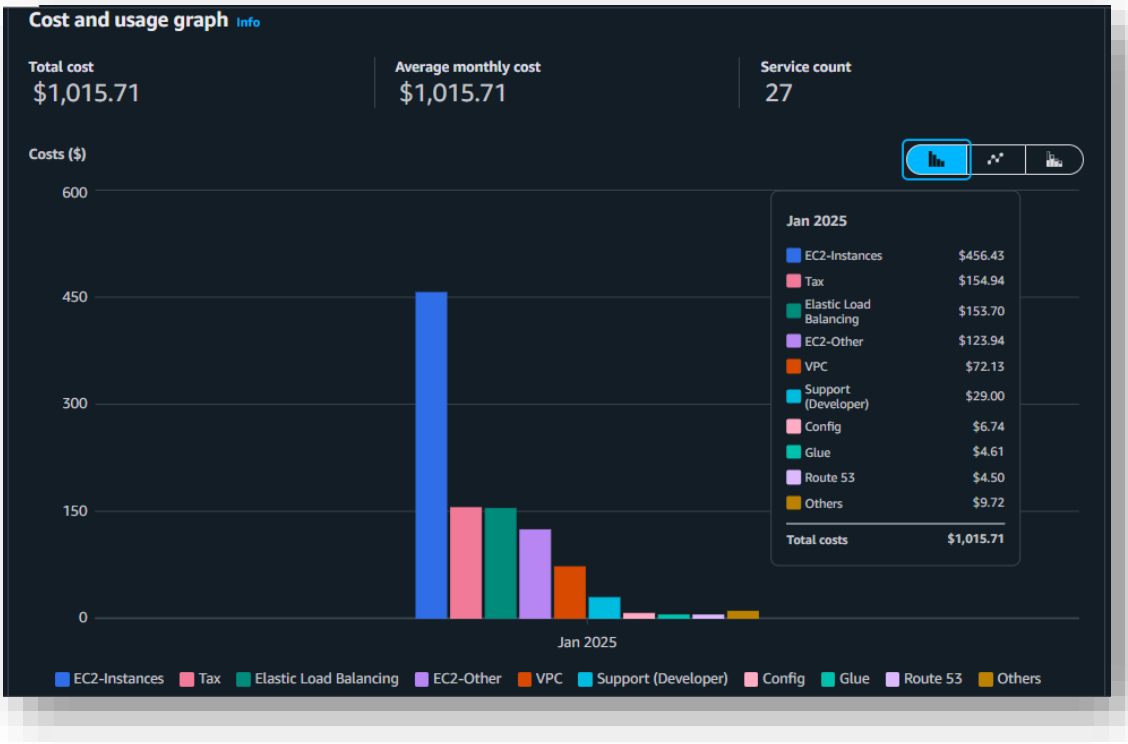


4. Set Up a CloudWatch Alarm



Describe how you configured notifications for the alarm:
chose a per-instance metric for my machine and chose the metric of CPU utilization. Then, configured it to work in 5 minutes periods and set a threshold to 70 as requiered. created an SNS topic and subscribed notifications to my personal Email address.

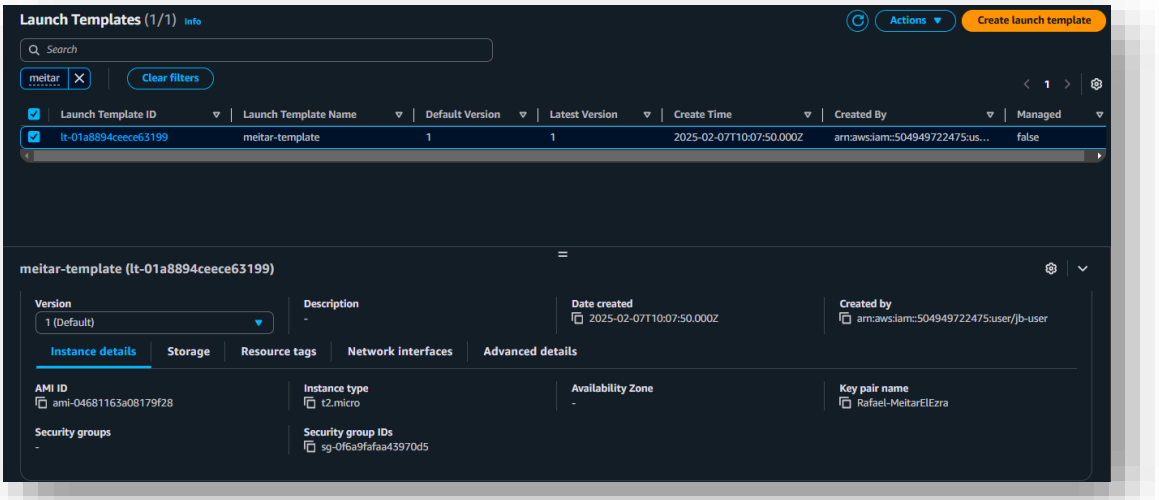
5. Identify AWS Billing Costs



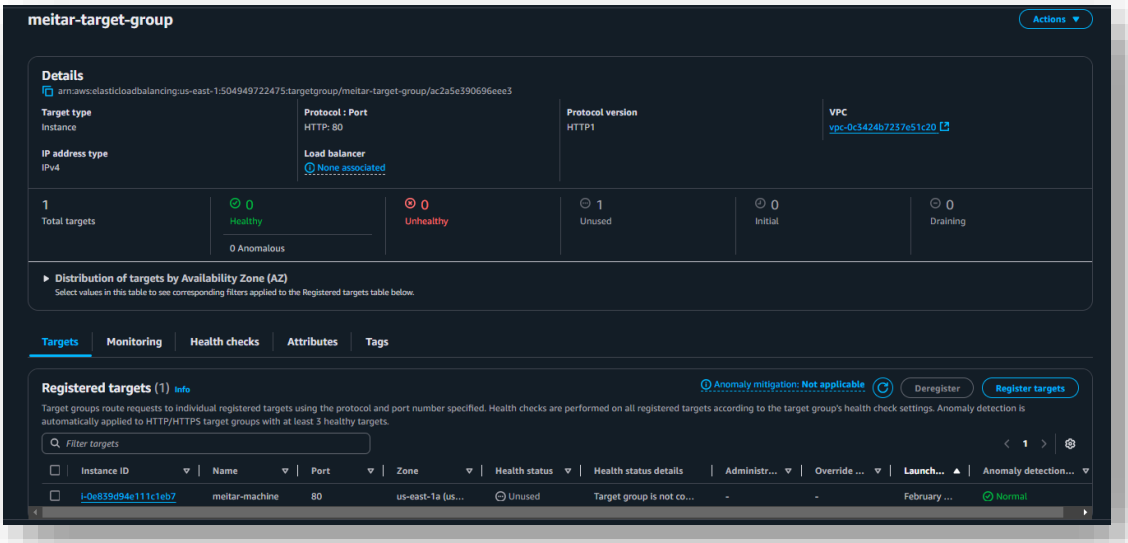
With EC2 instances as the largest cost, followed by taxes, elastic load balancing, and VPC, the total cost came to $1,015.71 for 27 services. There are no extra filters applied; costs are grouped by service. Right-sizing EC2 instances, reserving instances, and examining networking expenses are examples of possible efficiencies.

Section 3: Hands-on advanced

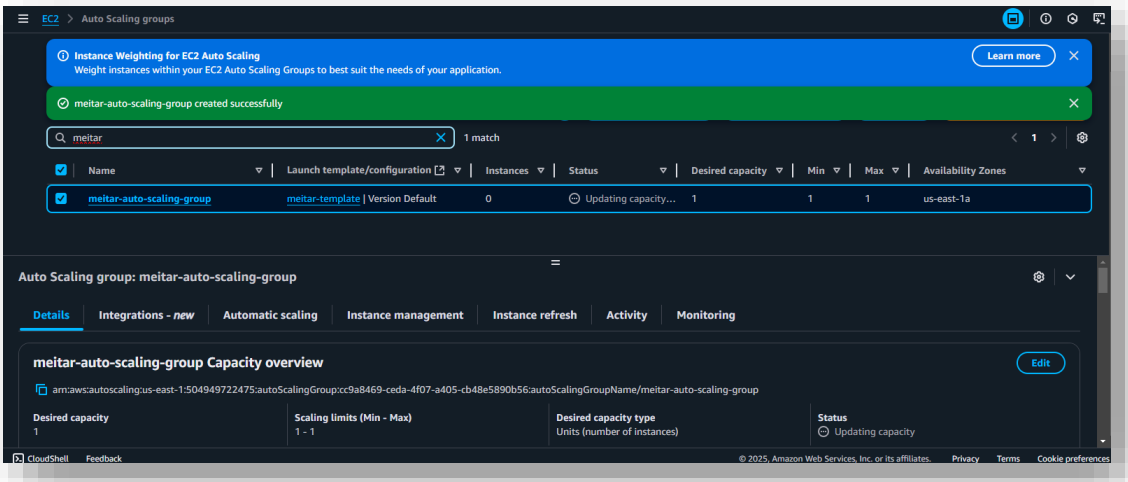
created launch template according to instructions



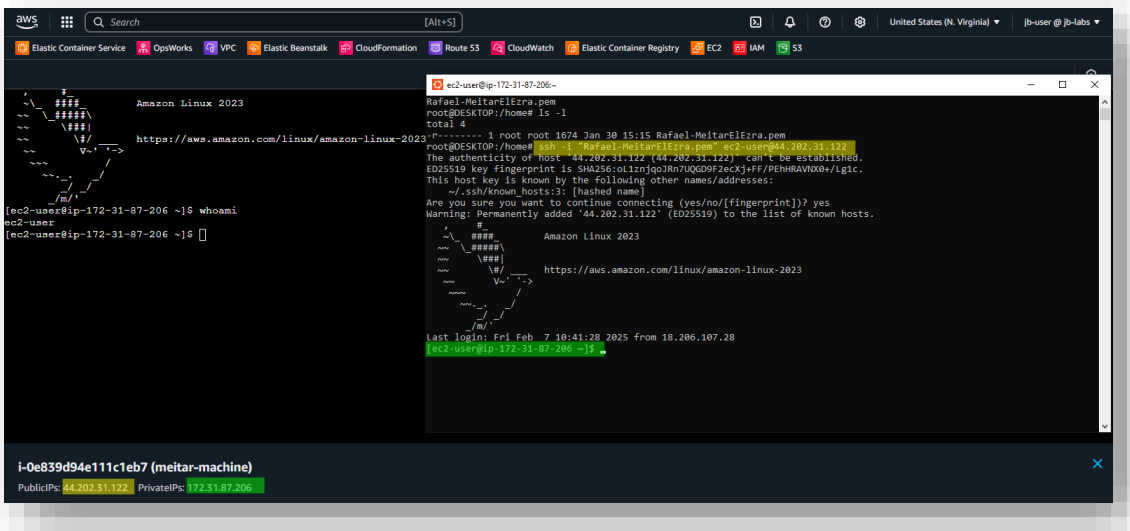
created target group with ec2 instance:



created auto scaling group using template with capacity of 1 instance:

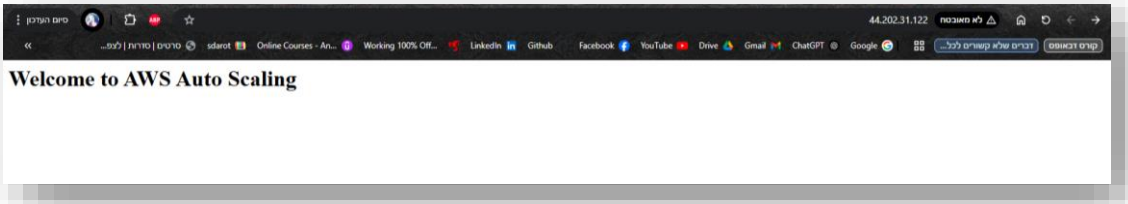


2. Connect to the EC2 Instance and Install Nginx



```
ec2-user@ip-172-31-87-206:~
[ec2-user@ip-172-31-87-206 ~]$ sudo yum update -y
Last metadata expiration check: 12:58:21 ago on Thu Feb  6 21:56:15 2025.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-87-206 ~]$ sudo yum install -y nginx
Last metadata expiration check: 12:58:48 ago on Thu Feb  6 21:56:15 2025.
Dependencies resolved.
================================================================================ Package
           Architecture       Version                          Repository          Size
================================================================================Installing:
 nginx                     x86_64           1:1.26.2-1.amzn2023.0.1             amazonlinux             33 k
Installing dependencies:
 generic-logos-httpd       noarch           18.0.0-12.amzn2023.0.3              amazonlinux             19 k
 gperftools-libs           x86_64           2.9.1-1.amzn2023.0.3                amazonlinux            308 k
 libunwind                 x86_64           1.4.0-5.amzn2023.0.2                amazonlinux             66 k
 nginx-core                x86_64           1:1.26.2-1.amzn2023.0.1             amazonlinux            670 k
 nginx-filesystem          noarch           1:1.26.2-1.amzn2023.0.1             amazonlinux            9.9 k
 nginx-mimetypes           noarch           2.1.49-3.amzn2023.0.3               amazonlinux             21 k

Transaction Summary
================================================================================Install  7 Packages

Total download size: 1.1 M
Installed size: 3.6 M
Downloading Packages:
(1/7): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm                523 kB/s |  19 kB     00:00
(2/7): libunwind-1.4.0-5.amzn2023.0.2.x86_64.rpm                            1.6 MB/s |  66 kB     00:00
(3/7): gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm                      5.7 MB/s | 308 kB     00:00
(4/7): nginx-core-1.26.2-1.amzn2023.0.1.x86_64.rpm                           19 MB/s | 670 kB     00:00
(5/7): nginx-filesystem-1.26.2-1.amzn2023.0.1.noarch.rpm                    428 kB/s | 9.9 kB     00:00
(6/7): nginx-1.26.2-1.amzn2023.0.1.x86_64.rpm                               744 kB/s |  33 kB     00:00
(7/7): nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch.rpm                     1.0 MB/s |  21 kB     00:00
--------------------------------------------------------------------------------Total
                                                    8.3 MB/s | 1.1 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                            1/1
  Running scriptlet: nginx-filesystem-1:1.26.2-1.amzn2023.0.1.noarch            1/7
  Installing       : nginx-filesystem-1:1.26.2-1.amzn2023.0.1.noarch            1/7
  Installing       : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch               2/7
  Installing       : libunwind-1.4.0-5.amzn2023.0.2.x86_64                      3/7
  Installing       : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64                4/7
  Installing       : nginx-core-1:1.26.2-1.amzn2023.0.1.x86_64                  5/7
  Installing       : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch          6/7
  Installing       : nginx-1:1.26.2-1.amzn2023.0.1.x86_64                       7/7
  Running scriptlet: nginx-1:1.26.2-1.amzn2023.0.1.x86_64                       7/7
  Verifying        : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch          1/7
  Verifying        : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64                2/7
  Verifying        : libunwind-1.4.0-5.amzn2023.0.2.x86_64                      3/7
  Verifying        : nginx-1:1.26.2-1.amzn2023.0.1.x86_64                       4/7
  Verifying        : nginx-core-1:1.26.2-1.amzn2023.0.1.x86_64                  5/7
  Verifying        : nginx-filesystem-1:1.26.2-1.amzn2023.0.1.noarch            6/7
  Verifying        : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch               7/7

Installed:
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch          gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
  libunwind-1.4.0-5.amzn2023.0.2.x86_64                      nginx-1:1.26.2-1.amzn2023.0.1.x86_64
  nginx-core-1:1.26.2-1.amzn2023.0.1.x86_64                  nginx-filesystem-1:1.26.2-1.amzn2023.0.1.noarch
  nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

Complete!
[ec2-user@ip-172-31-87-206 ~]$ echo "<h1>Welcome to AWS Auto Scaling</h1>" |sudo tee
<h1>Welcome to AWS Auto Scaling</h1>
[ec2-user@ip-172-31-87-206 ~]$ echo "<h1>Welcome to AWS Auto Scaling</h1>" | sudo tee /usr/share/nginx/html/index.html
<h1>Welcome to AWS Auto Scaling</h1>
[ec2-user@ip-172-31-87-206 ~]$ sudo systemctl start nginx
[ec2-user@ip-172-31-87-206 ~]$ sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[ec2-user@ip-172-31-87-206 ~]$ curl http://localhost:80
<h1>Welcome to AWS Auto Scaling</h1>
[ec2-user@ip-172-31-87-206 ~]$
```

the web page opens:



3. Access the Web Page via the Load Balancer

opening the LB DNS url in the browser: