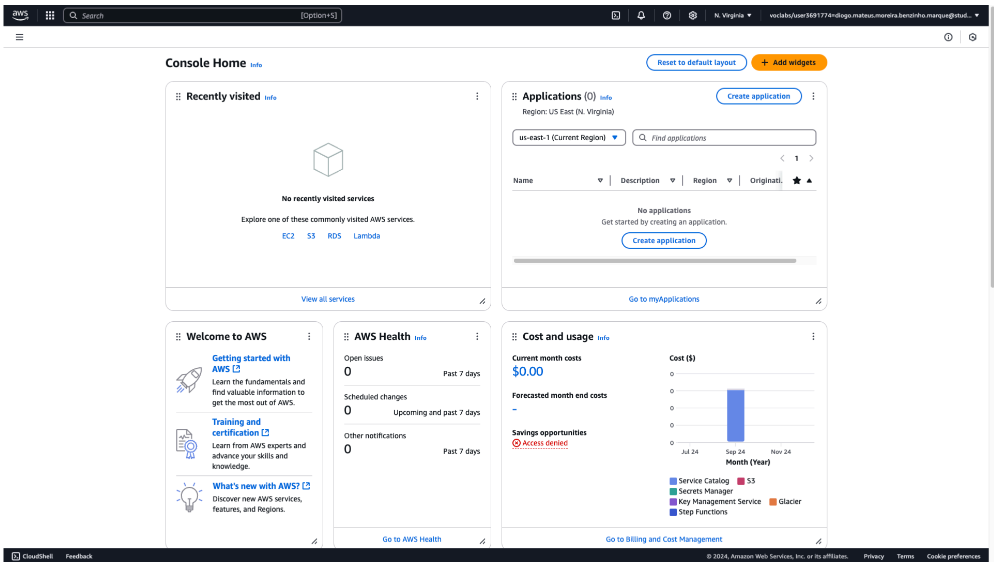
LS Lab Assignment: Amazon EC2*

1 Access


Console Home Info

Recently visited Info

No recently visited services

Explore one of these commonly visited AWS services.

EC2 S3 RDS Lambda

View all services

Applications (0) Info

Region: US East (N. Virginia)

No applications

Get started by creating an application.

Go to myApplications

Welcome to AWS

Getting started with AWS

Learn the fundamentals and find valuable information to get the most out of AWS.

Training and certification

Learn from AWS experts and advance your skills and knowledge.

What's new with AWS?

Discover new AWS services, features, and Regions.

AWS Health Info

Open issues	0	Past 7 days
Scheduled changes	0	Upcoming and past 7 days
Other notifications	0	Past 7 days

Go to AWS Health

Cost and usage Info

Current month costs $0.00

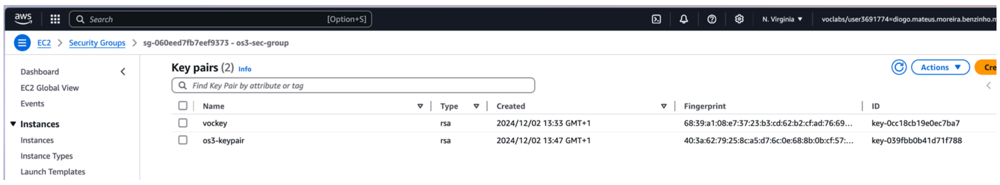
Forecasted month end costs -

Savings opportunities Access denied

Go to Billing and Cost Management

dashboard

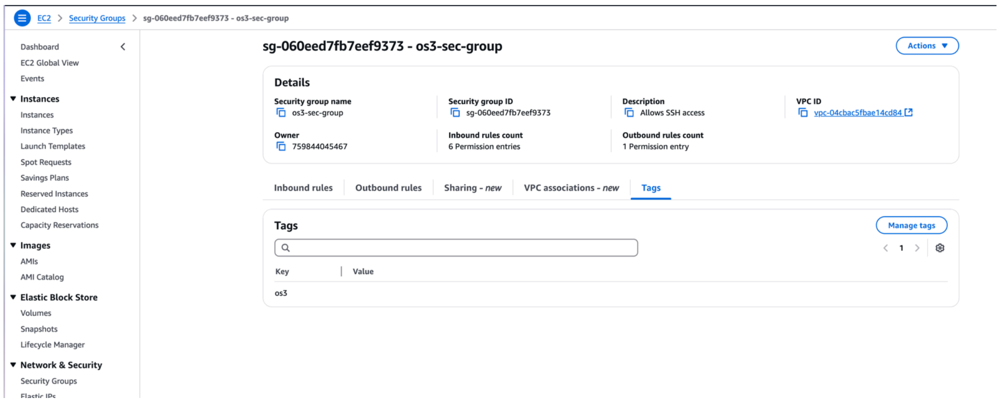
1 Key pair os3


Key pairs (2) Info

Name	Type	Created	Fingerprint	ID
vockey	rsa	2024/12/02 13:33 GMT+1	68:39:a1:08:e7:37:23:b3:cd:62:b2:cf:ad:76:69...	key-0cc18cb19e0ec7ba7
os3-keypair	rsa	2024/12/02 13:47 GMT+1	40:3a:62:79:25:8c:a5:d7:6c:0e:68:8b:0b:cf:57:...	key-039fbb0b41d71f788

key-pair.png

2 Network security group


sg-060eed7fb7eef9373 - os3-sec-group

Details

Security group name	Security group ID	Description	VPC ID
os3-sec-group	sg-060eed7fb7eef9373	Allows SSH access	vpc-04cbac5fbae14cd84

Owner	Inbound rules count	Outbound rules count
759844045467	6 Permission entries	1 Permission entry

Tags

Key	Value
os3	

network-security-group.png

2 Adding Instances

3 What is the difference between EBS and instance-store for VMs?

The key difference between **EBS** (Elastic Block Store) and **instance-store** lies in data persistence and use cases.

EBS is a persistent, durable block storage service that remains intact even when the EC2 instance is stopped or terminated. It is separate from the instance, allowing it to be detached and reattached to other instances. **EBS** supports snapshots, encryption, and offers customizable storage options, making it ideal for applications requiring long-term data storage, such as databases or file systems.

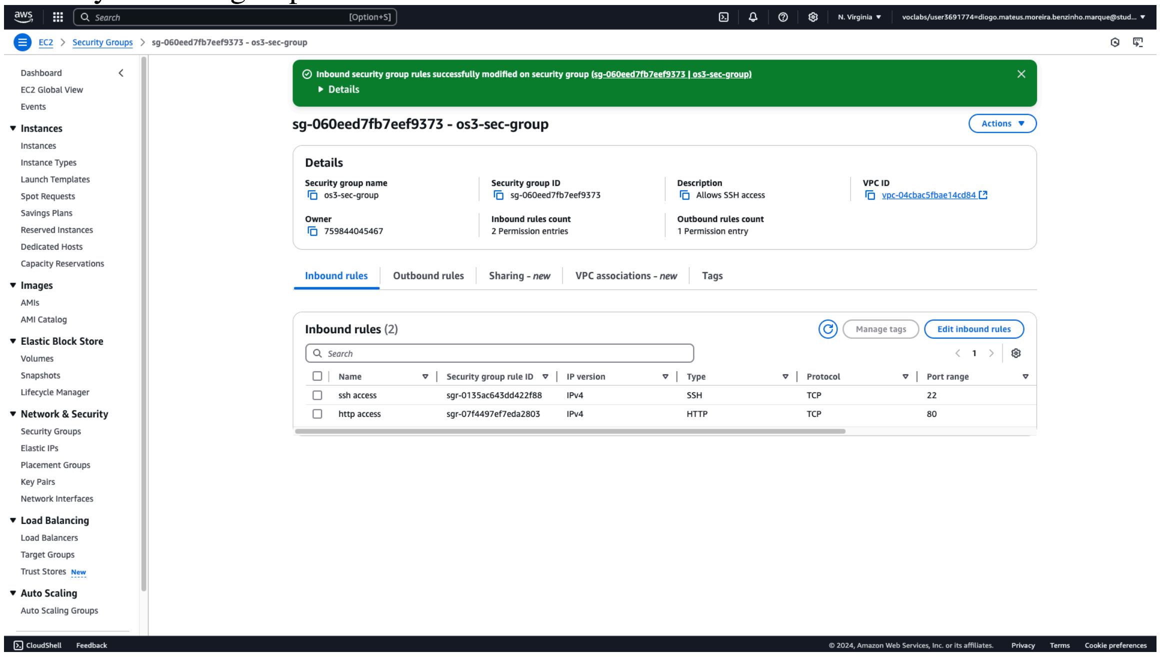
In contrast, **instance-store** is ephemeral storage physically attached to the host machine of the EC2 instance. Data in **instance-store** is temporary and lost when the instance is stopped or terminated. It does not support snapshots or customization and is typically used for temporary tasks like caching or data buffering that do not require persistence.

Feature	EBS	Instance-Store
Persistence	Persistent	Ephemeral
Attached to Instance	Detachable and re-attachable	Fixed to the instance
Data Loss on Stop/Restart	No	Yes
Scalability	Adjustable size	Fixed size based on instance type
Performance	Suitable for varied workloads	High-speed, local access
Snapshot Support	Yes	No
Backup Options	AWS Snapshots	None
Use Cases	Databases, application storage	Caching, temporary processing
Cost	Separate charge for volumes	Included in EC2 cost

In summary, **EBS** is suited for workloads requiring persistent and scalable storage, while **instance-store** is optimal for high-speed, temporary storage needs.

4 Add running instances to security group

Security network group inbound rules



Inbound security group rules successfully modified on security group (sg-060eed7fb7eef9373 | os3-sec-group)

sg-060eed7fb7eef9373 - os3-sec-group

Details

Security group name	Security group ID	Description	VPC ID
os3-sec-group	sg-060eed7fb7eef9373	Allows SSH access	vpc-04cbac5fbae14cd84

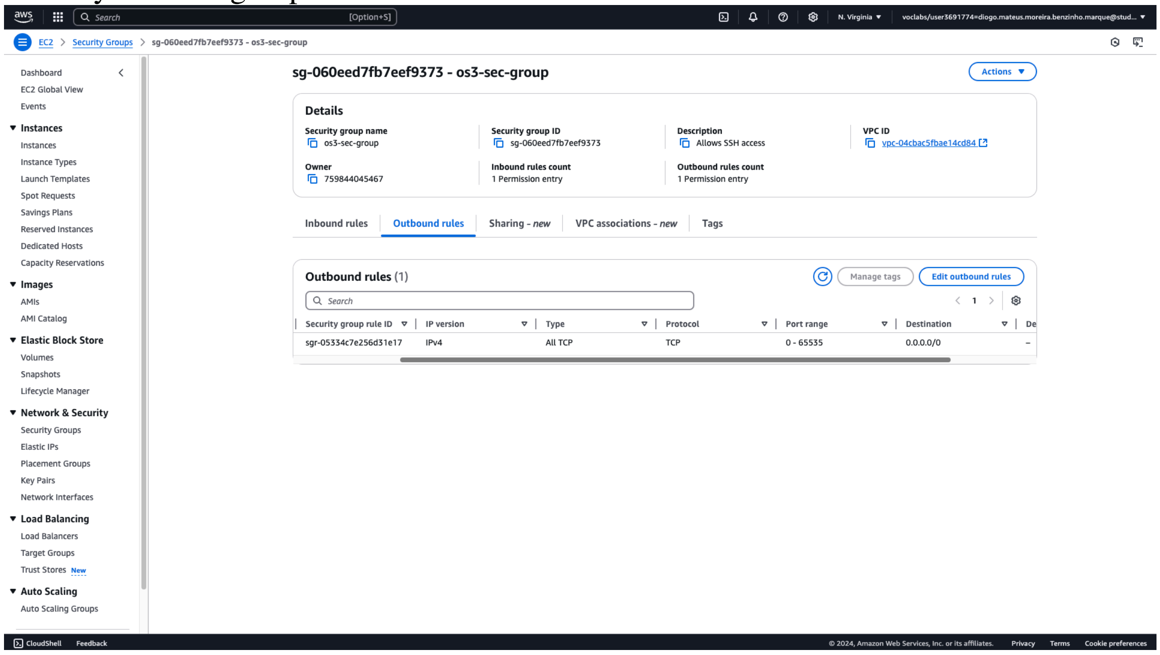
Owner	Inbound rules count	Outbound rules count
759844045467	2 Permission entries	1 Permission entry

Inbound rules | Outbound rules | Sharing - *new* | VPC associations - *new* | Tags

Inbound rules (2)

Name	Security group rule ID	IP version	Type	Protocol	Port range
ssh access	sgr-0135ac643dd422f88	IPv4	SSH	TCP	22
http access	sgr-07f4497ef7eda2803	IPv4	HTTP	TCP	80

Security network group outbound rules



sg-060eed7fb7eef9373 - os3-sec-group

Details

Security group name	Security group ID	Description	VPC ID
os3-sec-group	sg-060eed7fb7eef9373	Allows SSH access	vpc-04cbac5fbae14cd84

Owner	Inbound rules count	Outbound rules count
759844045467	1 Permission entry	1 Permission entry

Inbound rules | Outbound rules | Sharing - *new* | VPC associations - *new* | Tags

Outbound rules (1)

Security group rule ID	IP version	Type	Protocol	Port range	Destination	De
sgr-05334c7e256d31e17	IPv4	All TCP	TCP	0 - 65535	0.0.0.0/0	–

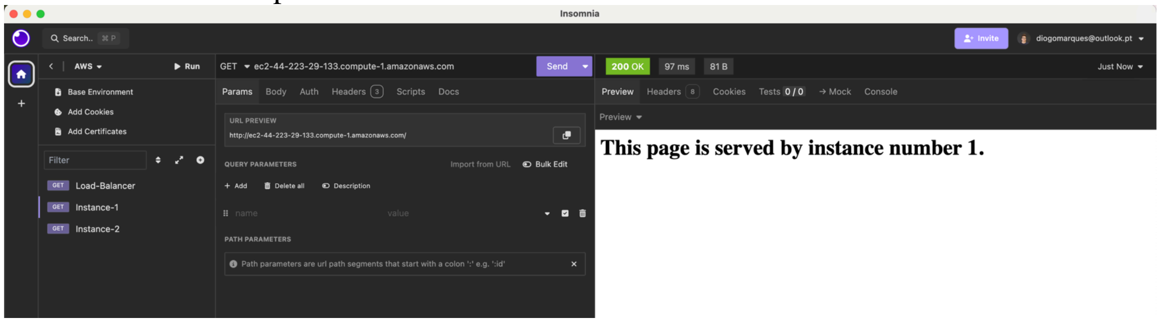
5 Create and Test the WebServers

Commands used to install and server the webserver for Linux machines.

```
sudo apt-get update
sudo apt-get install apache2 -y
sudo nano /var/www/html/index.html
cat /var/www/html/index.html
<html>
<body>
<h1>This page is served by instance number #.</h1>
</body>
</html>
```

6, 7 Request test done with AWS assigned DNS and measure response time

Instance 1. 97ms response time



GET ec2-44-223-29-133.compute-1.amazonaws.com

200 OK 97 ms 81 B

URL PREVIEW
http://ec2-44-223-29-133.compute-1.amazonaws.com/

QUERY PARAMETERS

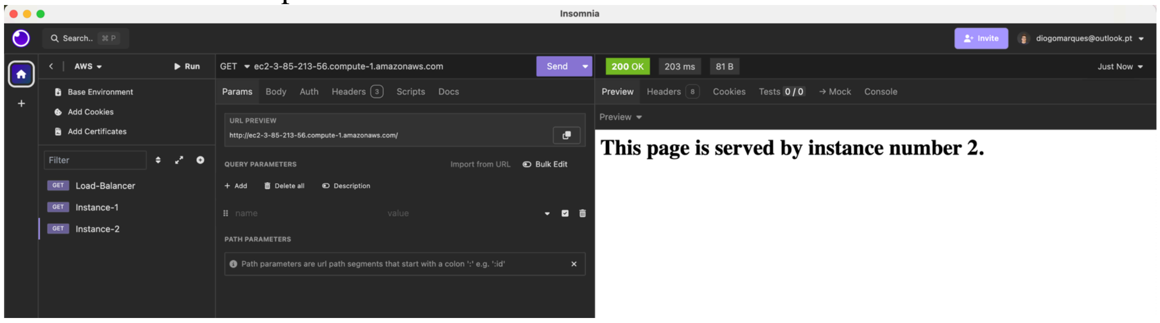
PATH PARAMETERS

Path parameters are url path segments that start with a colon ':' e.g. ':id'

Preview

This page is served by instance number 1.

Instance 2 203ms response time



GET ec2-3-85-213-56.compute-1.amazonaws.com

200 OK 203 ms 81 B

URL PREVIEW
http://ec2-3-85-213-56.compute-1.amazonaws.com/

QUERY PARAMETERS

PATH PARAMETERS

Path parameters are url path segments that start with a colon ':' e.g. ':id'

Preview

This page is served by instance number 2.

Create the script measure_response_time.py to perform the http requests.

```python
import httpx
import asyncio
import numpy as np
import time


# Function to calculate metrics from a list of response times
def calculate_metrics(response_times):
    """
    Calculates statistical metrics for a list of response times.

    Args:
        response_times (list): List of response times in seconds.

    Returns:
        dict: A dictionary containing metrics: average, median,
        minimum, maximum, and standard deviation.
    """
    return {
        "Average": np.mean(response_times),
        "Median": np.median(response_times),
        "Minimum": np.min(response_times),
        "Maximum": np.max(response_times),
        "Standard Deviation": np.std(response_times)
    }


# Asynchronous function to measure response times for multiple URLs n
times
async def measure_multiple_async_response_times(urls, n):
    """
    Measures response times for multiple URLs asynchronously, repeated
        n times.

    Args:
        urls (list): List of URLs to send GET requests to.
        n (int): Number of times to repeat the requests.

    Returns:
        list: A list of response times in seconds for all requests.
    """
    async with httpx.AsyncClient() as client:
        response_times = []

        for i in range(n):
            tasks = [client.get(url) for url in urls]  # Create a task
        for each URL
            start_time = time.time()
            responses = await asyncio.gather(*tasks)  # Wait for all
        tasks to complete
            end_time = time.time()

            # Calculate response times for this batch of URLs
            batch_time = end_time - start_time
            response_times.append(batch_time)
            print(f"Batch {i + 1}: Response time: {batch_time:.4f}
        seconds")

        return response_times

# Main function to demonstrate asynchronous response time measurement
if __name__ == "__main__":
        """
    Main function to measure response times for URLs run n times and
        calculate metrics.
    """
    # List of URLs to test
    urls = [
        "http://ec2-44-223-29-133.compute-1.amazonaws.com",
        # Ubuntu instance - 1 DNS
        "http://54.164.184.234",
        # Ubuntu instance - 1 IP
        "http://ec2-3-85-213-56.compute-1.amazonaws.com",
        # Ubuntu instance - 2 DNS
        "http://34.207.182.80",
        # Ubuntu instance - 2 IP
        "http://ec2-34-207-147-155.compute-1.amazonaws.com",
        # Windows instance
        "http://os3-load-balancer-1369570309.us-east-
        1.elb.amazonaws.com"   # Load Balancer
    ]

    # Number of times to repeat the requests
    n = 50

    # Measure response times asynchronously
    print("Asynchronous Response Time Measurement:")
    async_response_times =
        asyncio.run(measure_multiple_async_response_times(urls, n))

    # Calculate and print metrics for asynchronous requests
    async_metrics = calculate_metrics(async_response_times)
    print(f"\nAsynchronous Response Time Metrics for {n} Runs:")
    for metric, value in async_metrics.items():
        print(f"{metric}: {value:.4f} seconds")
```

There are the results for 50 requests to the Ubuntu Web server instances.

```
Asynchronous Response Time Metrics for 50 Runs:
Average: 0.1007 seconds
Median: 0.0983 seconds
Minimum: 0.0968 seconds
Maximum: 0.1988 seconds
Standard Deviation: 0.0142 seconds
```

Results for 50 requests.

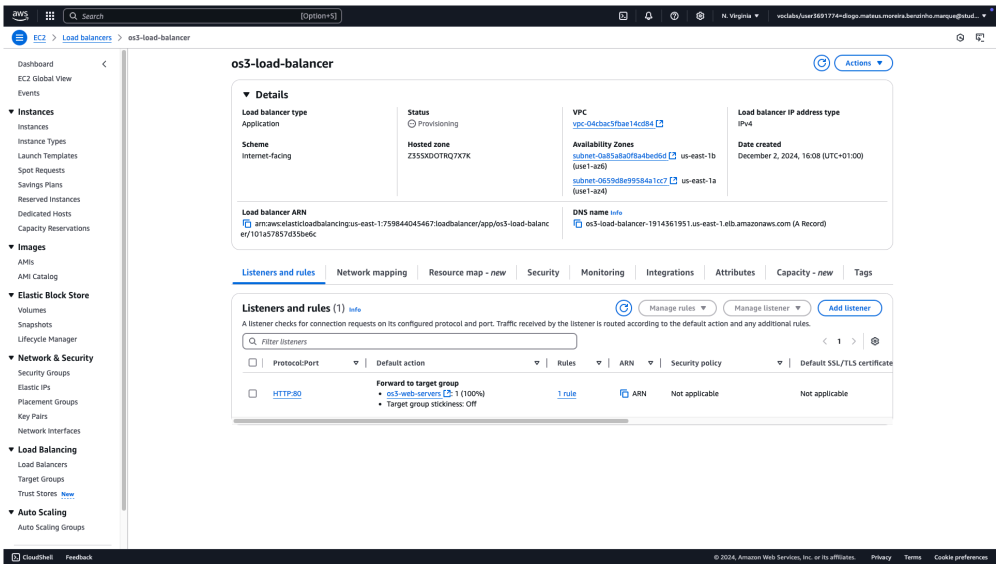
Metrics: - Average Response Time: 0.1007 seconds - Median: 0.0983 seconds - Minimum: 0.0968 seconds - Maximum: 0.1988 seconds Standard Deviation: 0.0142 seconds

Observations:

- The response times are highly consistent across batches, with the majority of batches averaging around 0.098 seconds.
- A single outlier in Batch 1 (0.1988 seconds) significantly increases the maximum response time.
- The low standard deviation indicates stable server performance without much variability between requests.

3 Optional: Load Balancing

Creation of Load Balancer



os3-load-balancer

Details

Load balancer type: Application
Status: Provisioning
VPC: vpc-04cbac5fbae14cd84
Load balancer IP address type: IPv4
Scheme: Internet-facing
Hosted zone: Z35SXDOTRQ7X7K
Availability Zones: subnet-0a85a8a0f8a4bed6d us-east-1b (use1-az6); subnet-0659d8e99584a1cc7 us-east-1a (use1-az4)
Date created: December 2, 2024, 16:08 (UTC+01:00)
Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:759844045467:loadbalancer/app/os3-load-balancer/101a57857d35be6c
DNS name: os3-load-balancer-1914361951.us-east-1.elb.amazonaws.com (A Record)

Listeners and rules | Network mapping | Resource map - *new* | Security | Monitoring | Integrations | Attributes | Capacity - *new* | Tags

Listeners and rules (1) Info

Manage rules | Manage listener | Add listener

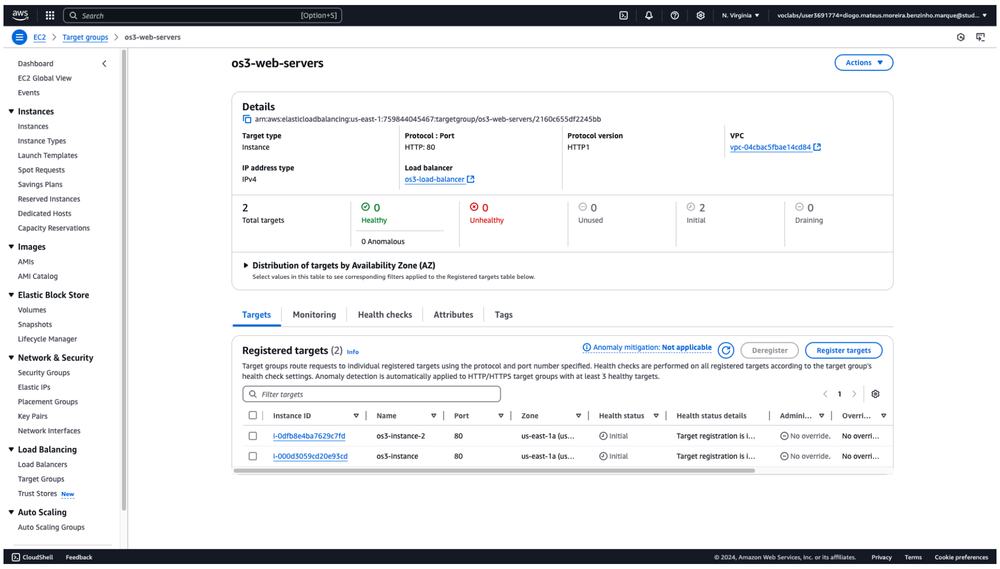
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate
HTTP:80	**Forward to target group** • os3-web-servers: 1 (100%) • Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable

load-balancer.png

Target groups


EC2 > Target groups > os3-web-servers
Actions
Targets | Monitoring | Health checks | Attributes | Tags
Deregister
Register targets

os3-web-servers

Details
arn:aws:elasticloadbalancing:us-east-1:759844045467:targetgroup/os3-web-servers/2160c655df2245bb

Target type	Protocol : Port	Protocol version	VPC
Instance	HTTP: 80	HTTP1	vpc-04cbac5fbae14cd84

IP address type	Load balancer
IPv4	os3-load-balancer

2	0	0	0	2	0
Total targets	Healthy	Unhealthy	Unused	Initial	Draining
	0 Anomalous				

Distribution of targets by Availability Zone (AZ)
Select values in this table to see corresponding filters applied to the Registered targets table below.

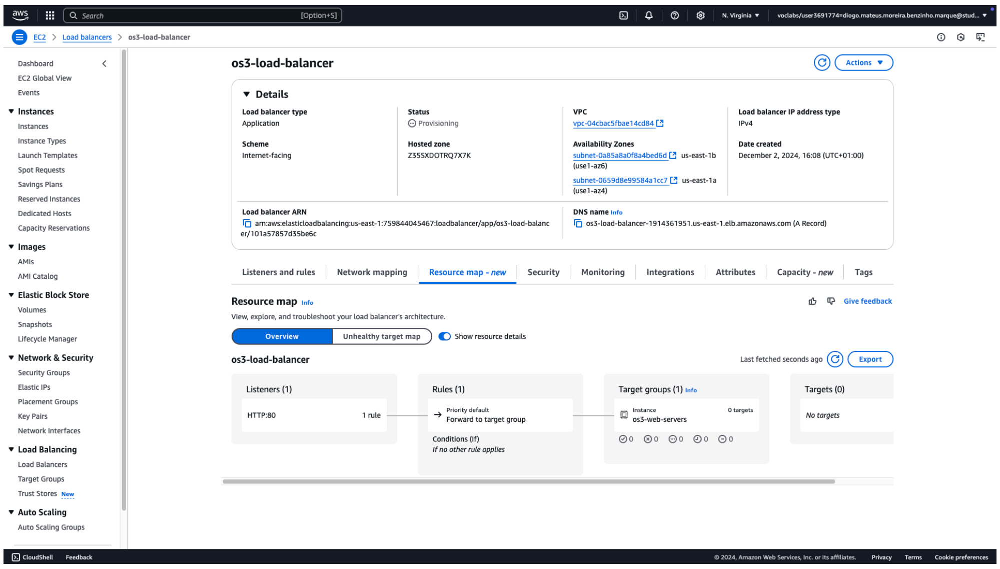
Registered targets (2) Info — Anomaly mitigation: Not applicable

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Instance ID	Name	Port	Zone	Health status	Health status details	Admini...	Overri...
i-0dfb8e4ba7629c7fd	os3-instance-2	80	us-east-1a (us...	Initial	Target registration is i...	No override.	No overri...
i-000d3059cd20e93cd	os3-instance	80	us-east-1a (us...	Initial	Target registration is i...	No override.	No overri...

load-balancer-target-group.png

Confirmation with Resource Map view tool


EC2 > Load balancers > os3-load-balancer
Actions
Listeners and rules | Network mapping | Resource map - new | Security | Monitoring | Integrations | Attributes | Capacity - new | Tags
Give feedback
Overview
Unhealthy target map
Show resource details
Export

os3-load-balancer

Details

Load balancer type	Status	VPC	Load balancer IP address type
Application	Provisioning	vpc-04cbac5fbae14cd84	IPv4

Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z35SXDOTRQ7X7K	subnet-0a85a8a0f8a4bed6d us-east-1b (use1-az6); subnet-0659d8e99584a1cc7 us-east-1a (use1-az4)	December 2, 2024, 16:08 (UTC+01:00)

Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:759844045467:loadbalancer/app/os3-load-balancer/101a57857d35be6c

DNS name: os3-load-balancer-1914361951.us-east-1.elb.amazonaws.com (A Record)

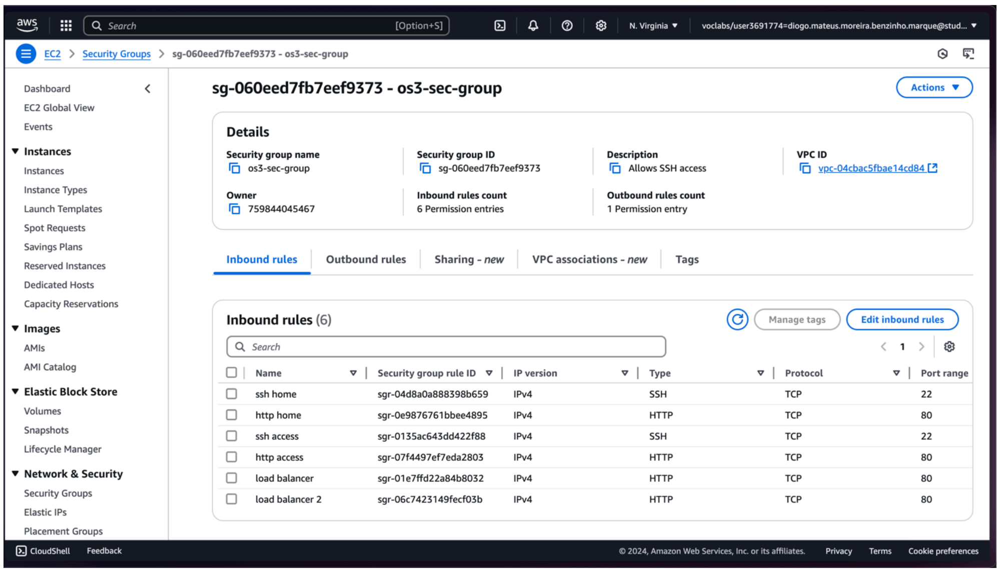
Resource map Info
View, explore, and troubleshoot your load balancer's architecture.

os3-load-balancer — Last fetched seconds ago

Listeners (1): HTTP:80 — 1 rule
Rules (1): Priority default — Forward to target group; Conditions (If): If no other rule applies
Target groups (1): Instance os3-web-servers — 0 targets; 0 0 0 0 0
Targets (0): No targets

load-balencer-2.png

Add load balancer to network security group


EC2 > Security Groups > sg-060eed7fb7eef9373 - os3-sec-group
Actions
Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags
Manage tags
Edit inbound rules

sg-060eed7fb7eef9373 - os3-sec-group

Details

Security group name	Security group ID	Description	VPC ID
os3-sec-group	sg-060eed7fb7eef9373	Allows SSH access	vpc-04cbac5fbae14cd84

Owner	Inbound rules count	Outbound rules count
759844045467	6 Permission entries	1 Permission entry

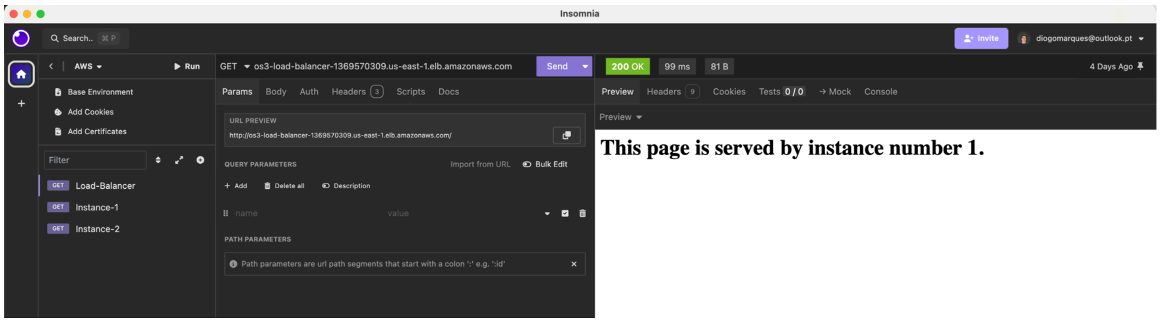
Inbound rules (6)

Name	Security group rule ID	IP version	Type	Protocol	Port range
ssh home	sgr-04d8a0a888398b659	IPv4	SSH	TCP	22
http home	sgr-0e9876761bbee4895	IPv4	HTTP	TCP	80
ssh access	sgr-0135ac643dd422f88	IPv4	SSH	TCP	22
http access	sgr-07f4497ef7eda2803	IPv4	HTTP	TCP	80
load balancer	sgr-01e7ffd22a84b8032	IPv4	HTTP	TCP	80
load balancer 2	sgr-06c7423149fecf03b	IPv4	HTTP	TCP	80

load-balancer-security-group.png

8 Test load balancer with Http request to DNS

Instance 1



GET os3-load-balancer-1369570309.us-east-1.elb.amazonaws.com Send

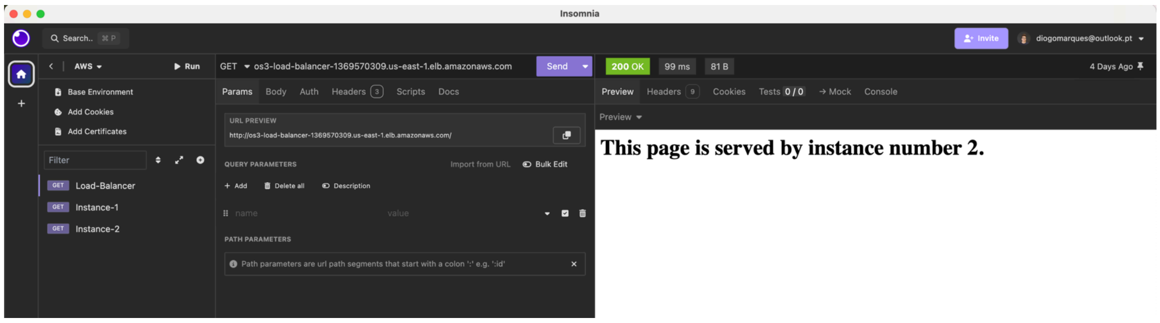
200 OK 99 ms 81 B

URL PREVIEW http://os3-load-balancer-1369570309.us-east-1.elb.amazonaws.com/

Path parameters are url path segments that start with a colon ':' e.g. ':id'

This page is served by instance number 1.

Instance 2



GET os3-load-balancer-1369570309.us-east-1.elb.amazonaws.com Send

200 OK 99 ms 81 B

URL PREVIEW http://os3-load-balancer-1369570309.us-east-1.elb.amazonaws.com/

Path parameters are url path segments that start with a colon ':' e.g. ':id'

This page is served by instance number 2.

9 - 11 50 Request results (measure the time)

```
Asynchronous Response Time Metrics for 50 Runs:
Average: 0.1016 seconds
Median: 0.0989 seconds
Minimum: 0.0966 seconds
Maximum: 0.2206 seconds
Standard Deviation: 0.0170 seconds
```

Metrics: - Average Response Time: 0.1016 seconds - Median: 0.0989 seconds - Minimum: 0.0966 seconds - Maximum: 0.2206 seconds Standard Deviation: 0.0170 seconds

Observations:

- The response times are similar to the scenario without the load balancer but show slightly more variability (higher standard deviation).
- Batch 1 has a significantly higher response time (0.2206 seconds), likely due to load balancer initialization or a cold start.
- Subsequent batches stabilize, with times clustering around 0.098-0.100 seconds, similar to the results without the load balancer.

Comparison

Performance Consistency:

Without the load balancer, the system exhibits slightly more consistent response times (lower standard deviation). With the load balancer, there is marginally higher variability, likely due to the overhead introduced by the load balancer or differences in how requests are distributed to backend servers.

Initial Latency:

The first batch with the load balancer exhibits a noticeable delay, suggesting some additional processing or initialization overhead when the load balancer starts handling requests.

Distribution:

The similarity in average and median response times suggests that the load balancer effectively distributes requests across backend servers. There is no significant evidence of imbalanced load distribution or server bottlenecks, indicating that the load balancer is performing its function adequately.

Hypothesis Confirmation

Based on the data, the load balancer likely employs the Round-Robin technique or a similar balanced distribution strategy. This ensures a near-equal workload across servers, resulting in comparable response times. The marginal increase in variability and overhead could be attributed to: Load balancer processing. Slight differences in backend server performance.

10 How Does Continuous Requests Influence the Load Balancer?

Continuous requests to the load balancer can:

Increase the CPU and memory usage on the load balancer due to constant traffic routing. Expose inefficiencies in request distribution if the load balancer is not configured correctly (e.g., Round-Robin, Least Connections, or Weighted Round-Robin). Lead to possible queuing or latency issues, depending on the capacity of the backend servers and the load balancer’s processing limits.

11 Measure the HTTP response time both on the load balancer as well as on the separate web servers. (9)

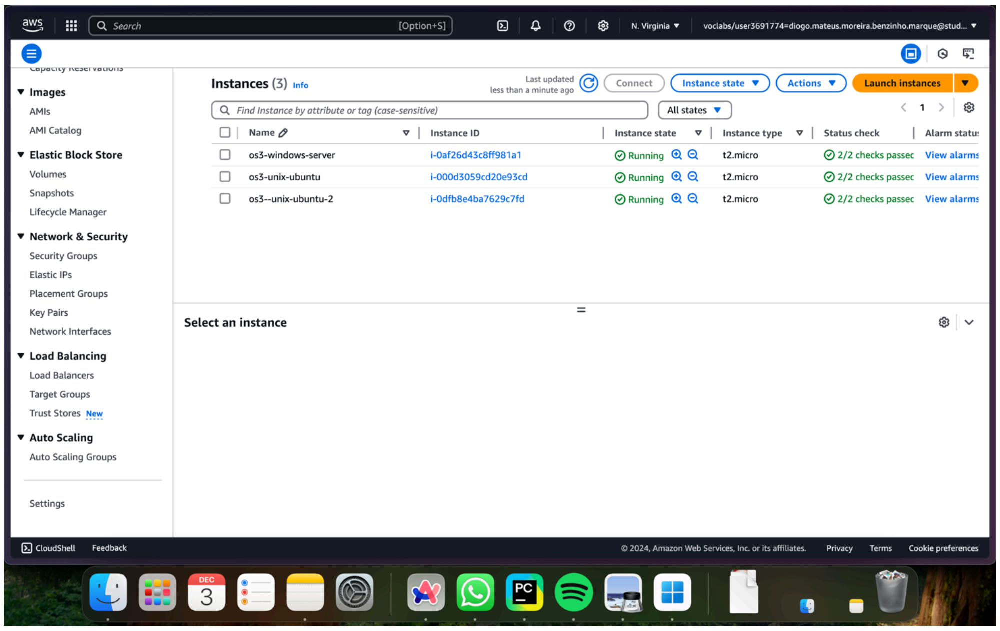
12 Conclusion from Results

From the collected data:

The load balancer and backend servers are handling the load efficiently. The balancing strategy is likely effective.

Windows Server IIS Setup

Dashboard



Instances (3) Info

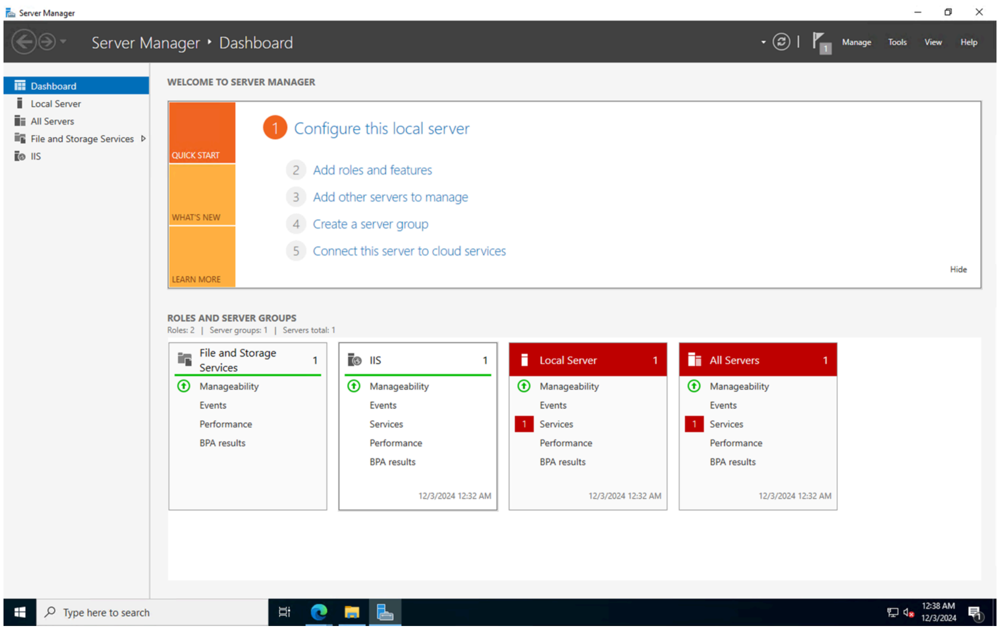
Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm statu
os3-windows-server	i-0af26d43c8ff981a1	Running	t2.micro	2/2 checks passec	View alarms
os3-unix-ubuntu	i-000d3059cd20e93cd	Running	t2.micro	2/2 checks passec	View alarms
os3--unix-ubuntu-2	i-0dfb8e4ba7629c7fd	Running	t2.micro	2/2 checks passec	View alarms

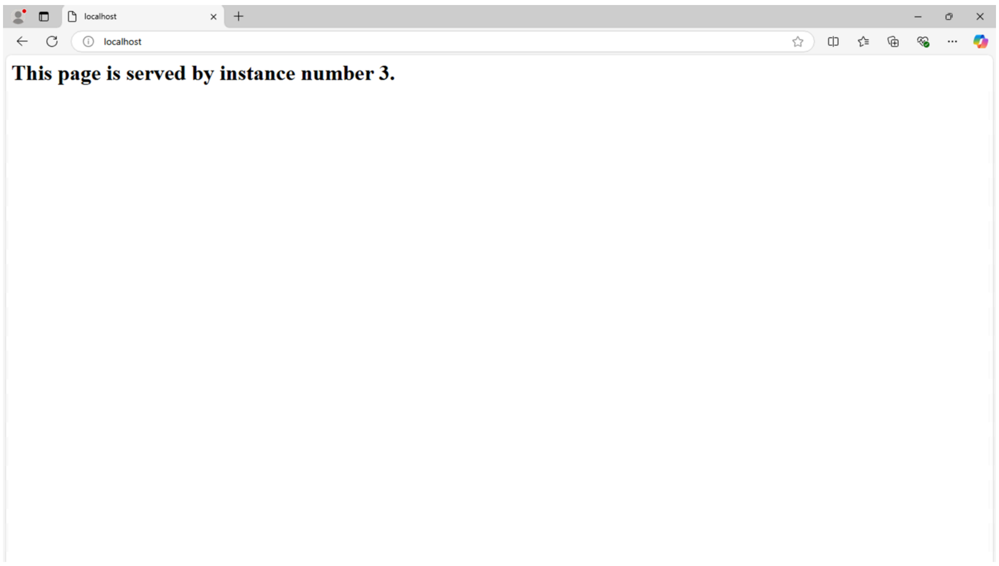
dashboard-windows.png

Setup



windows-server-iis.png

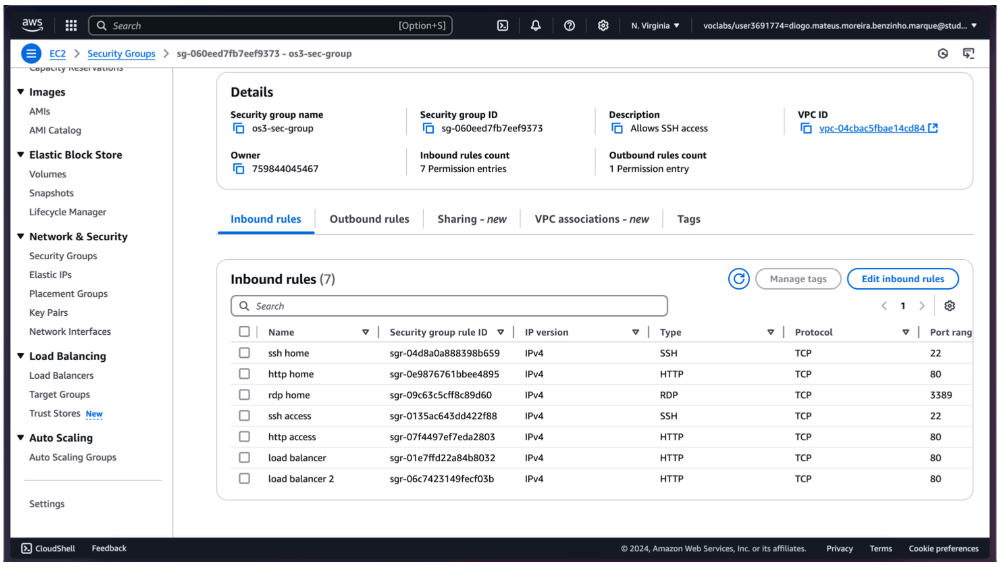
HTTP Validation on server



This page is served by instance number 3.

windows-iis.png

Add to security group to be available on the load balancer



Details

Security group name	Security group ID	Description	VPC ID
os3-sec-group	sg-060eed7fb7eef9373	Allows SSH access	vpc-04cbac5fbae14cd84
Owner	Inbound rules count	Outbound rules count	
759844045467	7 Permission entries	1 Permission entry	

Inbound rules (7)

Name	Security group rule ID	IP version	Type	Protocol	Port rang
ssh home	sgr-04d8a0a888398b659	IPv4	SSH	TCP	22
http home	sgr-0e9876761bbee4895	IPv4	HTTP	TCP	80
rdp home	sgr-09c63c5cff8c89d60	IPv4	RDP	TCP	3389
ssh access	sgr-0135ac643dd422f88	IPv4	SSH	TCP	22
http access	sgr-07f4497ef7eda2803	IPv4	HTTP	TCP	80
load balancer	sgr-01e7ffd22a84b8032	IPv4	HTTP	TCP	80
load balancer 2	sgr-06c7423149fecf03b	IPv4	HTTP	TCP	80

security-group-windows.png

13. Advantages/Disadvantages of a Mixed Setup

Advantages: - Increased Diversity: Mixing Windows and Ubuntu servers introduces redundancy, reducing downtime risk if one type of server fails.

- Flexibility: A mixed environment allows leveraging platform-specific features (e.g., IIS on Windows for .NET applications).
- Improved Scalability: Distributing load across different server types can enhance overall system scalability.

Disadvantages: - Configuration Complexity: Managing and maintaining a heterogeneous environment can be more challenging.

- Potential Performance Imbalance: Differences in server performance may lead to uneven load distribution, requiring weighted load balancing.

- Software Compatibility: Additional work may be needed to ensure compatibility across platforms.

4 Optional: Security

14. Behavioral Change with Three Servers

Load balancing including windows server

```
Asynchronous Response Time Metrics for 50 Runs:
Average: 0.0973 seconds
Median: 0.0953 seconds
Minimum: 0.0935 seconds
Maximum: 0.2010 seconds
Standard Deviation: 0.0149 seconds
```

Response Times with 3 Backend Servers (Ubuntu + IIS)

Metrics: - Average Response Time: 0.0973 seconds - Median: 0.0953 seconds - Minimum: 0.0935 seconds - Maximum: 0.2010 seconds Standard Deviation: 0.0149 seconds

Observations: - The average response time and median response time are consistent with previous setups involving two Ubuntu servers, indicating that the load balancer effectively distributes requests to the three servers. - The maximum response time (0.2010 seconds) is higher than most other batches, suggesting occasional delays, potentially due to:

- Load balancer overhead. Performance differences between the IIS server and the Ubuntu servers.
- Performance Consistency:
  - The low standard deviation (0.0149 seconds) shows that response times are stable across most batches.
  - The slight variability could stem from differences in how IIS handles requests compared to Ubuntu servers running a similar page.

Advantages of Adding a Third Server

Increased Redundancy:

The addition of an IIS server provides another backend server, reducing the risk of downtime if one server fails. The load balancer now has more options to distribute traffic, potentially improving availability and resilience.

Improved Load Distribution:

With three servers, the workload is more evenly distributed, reducing the likelihood of overloading any single server.

Disadvantages of a Mixed Setup

Performance Variability:

If the IIS server has different hardware resources or software configurations, it could handle requests more slowly or differently compared to Ubuntu servers. This variability might increase the response time for some batches, as evidenced by the outlier maximum response time.

Complexity:

Managing a mixed environment adds complexity in terms of maintenance, monitoring, and ensuring consistent behavior across platforms. IIS might require different tools or settings compared to Ubuntu, which could increase operational overhead.

Compatibility Issues:

While serving static content is straightforward, mixed setups can face compatibility issues with dynamic or application-specific features. Behavioral Changes with Three Servers

Load Balancer Behavior:

The load balancer successfully included the IIS server in its rotation, as evidenced by consistent response times across batches. The absence of significant performance degradation suggests that the load balancer’s algorithm (likely Round-Robin or Least Connections) distributes traffic effectively.

Impact of IIS Server:

The IIS server seems to handle requests comparably to the Ubuntu servers for static content, given the similar average and median response times. Occasional higher response times might indicate slight differences in how requests are processed or resource availability. Conclusion

Effectiveness of the Load Balancer:

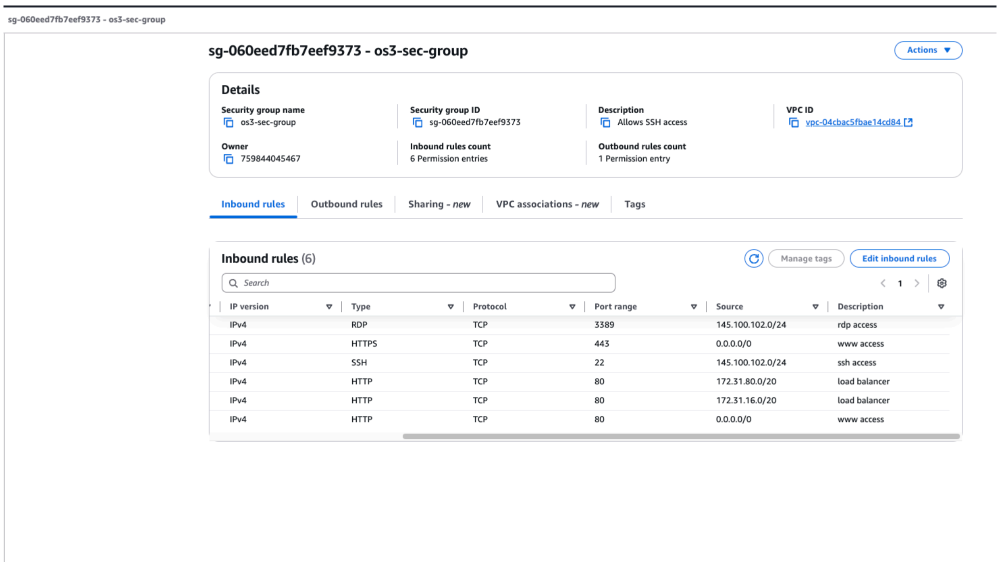
The load balancer continues to distribute requests efficiently, even with three servers. The results suggest that adding the IIS server did not introduce significant issues in handling requests or response time consistency.

Recommendation for Mixed Environments:

Mixed setups work well for simple static content, but additional performance monitoring is recommended to identify and address variability between server types. For more complex setups, consider using weighted balancing to account for differences in server performance.

15 Modify the Security Group you created in Question 2 for your web servers with reasonable inbound traffic rules

Added HTTP, HTTPS/SSL inbound rules worldwide, RDP/SSH restricted to OS3 network and Load balancer IPs for health checks



sg-060eed7fb7eef9373 - os3-sec-group

sg-060eed7fb7eef9373 - os3-sec-group

Details

Security group name	Security group ID	Description	VPC ID
os3-sec-group	sg-060eed7fb7eef9373	Allows SSH access	vpc-04cbac5fbae14cd84
Owner	Inbound rules count	Outbound rules count	
759844045467	6 Permission entries	1 Permission entry	

Inbound rules (6)

IP version	Type	Protocol	Port range	Source	Description
IPv4	RDP	TCP	3389	145.100.102.0/24	rdp access
IPv4	HTTPS	TCP	443	0.0.0.0/0	www access
IPv4	SSH	TCP	22	145.100.102.0/24	ssh access
IPv4	HTTP	TCP	80	172.31.80.0/20	load balancer
IPv4	HTTP	TCP	80	172.31.16.0/20	load balancer
IPv4	HTTP	TCP	80	0.0.0.0/0	www access

security_group

5 Budgeting

16 If you were to build these infrastructure for a small-to-medium sized enterprise, how much should the company budget for 1 year for cloud serv

Based on the AWS cost estimate and infrastructure setup, the company should budget approximately $419.40 USD for the first year for cloud services.

This budget will cover the following components:

- 2 EC2 Micro Instances running Ubuntu 24.04 for hosting web servers.
- 1 EC2 Micro Instance running Microsoft Windows Server 2022 for additional service hosting.
- 1 Application Load Balancer to distribute incoming traffic across the EC2 instances.

Breakdown of Estimated Annual Cost:

- **Monthly Cost**: $34.95 USD.
- **Total Annual Cost**: $419.40 USD.

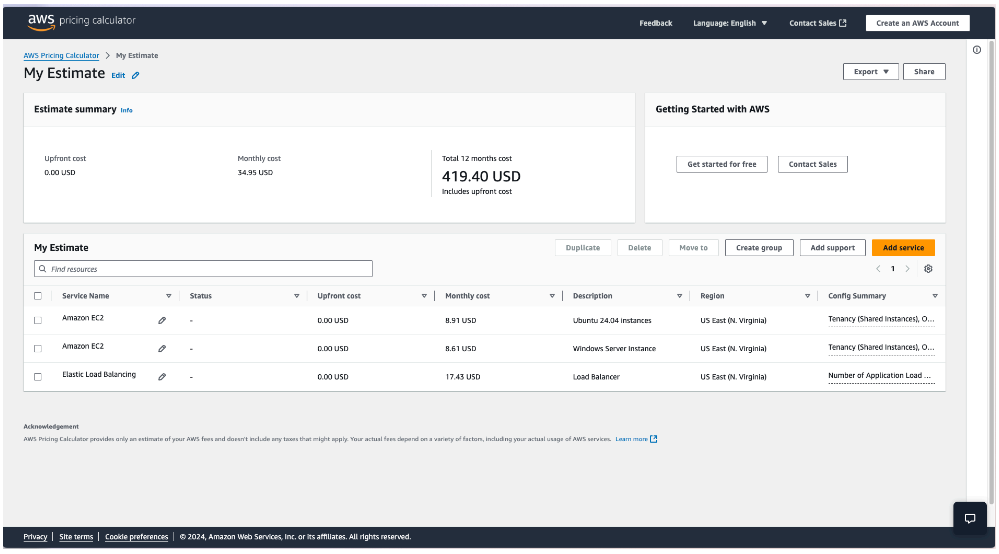
This budget is calculated based on on-demand pricing with no upfront costs and assumes consistent usage throughout the year.

Considerations:

- **Scalability**: The enterprise may need to adjust the budget if more instances or additional resources (e.g., storage, backup, or data transfer) are needed.
- **Flexibility**: On-demand pricing offers flexibility but is generally higher than reserved instances, so if the company's usage becomes more predictable, they could consider

reserved instances or savings plans in the future to reduce costs. This estimate provides a reasonable starting point for a small-to-medium-sized enterprise running standard web infrastructure on AWS.

17 AWS Pricing Calculator can generate a persistent link for you to refer to in the future. Provide your estimate's public link for possible discussions with your TAs.



My Estimate Edit

Estimate summary Info

Upfront cost 0.00 USD

Monthly cost 34.95 USD

Total 12 months cost 419.40 USD Includes upfront cost

Getting Started with AWS

My Estimate

Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary
Amazon EC2	-	0.00 USD	8.91 USD	Ubuntu 24.04 instances	US East (N. Virginia)	Tenancy (Shared Instances), O...
Amazon EC2	-	0.00 USD	8.61 USD	Windows Server Instance	US East (N. Virginia)	Tenancy (Shared Instances), O...
Elastic Load Balancing	-	0.00 USD	17.43 USD	Load Balancer	US East (N. Virginia)	Number of Application Load ...

Acknowledgement

AWS Pricing Calculator provides only an estimate of your AWS fees and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services. Learn more

Privacy | Site terms | Cookie preferences | © 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

estimation-costs.png

Public link to AWS Budget:

6 Virtual Data Center

18 Do you think this is feasible with the current AWS services?

Yes, migrating all the infrastructure—storage, networking, servers (web, application, database), and security services—to the Amazon Cloud is not only feasible but also a common approach for many businesses. AWS offers a wide range of services that support these needs, allowing enterprises to move their entire infrastructure to the cloud.

19 Briefly explain how you would use each AWS service to implement the sysadmin's plan.

To implement the plan of migrating all infrastructure (storage, network, servers, and security) to AWS, here's a brief explanation of how each service can be used:

1. Storage

- **Amazon S3 (Simple Storage Service)**:
  Use for storing unstructured data, backups, and static content (e.g., images, logs, documents). S3 provides scalable, durable object storage with low latency.
- **Amazon EBS (Elastic Block Store)**:
  Use for persistent block storage attached to EC2 instances. EBS is ideal for databases or file systems where low-latency access is crucial.

- **Amazon EFS (Elastic File System)**:
  Use for shared file storage across EC2 instances. Ideal for applications requiring a common file system across multiple servers.

2. Networking

- **Amazon VPC (Virtual Private Cloud)**:
  Create isolated virtual networks within AWS. Use VPC to segment your infrastructure, set up private subnets, configure route tables, and manage network traffic.

- **AWS Direct Connect**:
  Establish a dedicated network connection from your on-premises data center to AWS. Useful for companies that need high throughput or consistent network performance.

- **Amazon Route 53**:
  Use as a DNS service to manage domain names and route internet traffic to various AWS resources (e.g., EC2 instances, load balancers).

3. Servers (Web, Application, Database)

- **Amazon EC2 (Elastic Compute Cloud)**:
  Launch and manage scalable compute instances for web, application, and database servers. EC2 provides a wide range of instance types suitable for different workloads, and you can adjust resources as needed.

  - **Web Servers**: Use EC2 instances running Apache or Nginx to serve web pages.
  - **Application Servers**: EC2 can also host application servers that run backend services (e.g., Node.js, Python, or Java).
  - **Database Servers**: EC2 instances can be used to deploy and manage your own database systems (e.g., MySQL, PostgreSQL, etc.), or you can use managed database services (see below).

- **AWS Lambda**:
  For serverless computing. Use Lambda for running small pieces of code (functions) triggered by events, without the need to manage servers.

4. Databases

- **Amazon RDS (Relational Database Service)**:
  Use RDS for managed relational databases like MySQL, PostgreSQL, Oracle, or SQL Server. RDS simplifies database management by automating tasks like backups, patching, and scaling.

- **Amazon Aurora**:
  A high-performance, fully managed relational database compatible with MySQL and PostgreSQL. Aurora is designed for high availability and read scalability.

- **Amazon DynamoDB**:
  Use for scalable, low-latency NoSQL database needs. Ideal for handling large volumes of unstructured data, and it's fully managed with built-in security and scalability.

5. Security

- **AWS Identity and Access Management (IAM)**:
  Use IAM to manage users, roles, and permissions across your AWS environment. Define who can access what resources and enforce security policies.

- **Security Groups and Network ACLs**:
  Use these to control inbound and outbound traffic to/from your EC2 instances, ensuring that only authorized users and systems can access your web and application servers.

- **AWS WAF (Web Application Firewall)**:
  Protect your applications from common web exploits by using AWS WAF to filter and monitor HTTP traffic.

- **AWS Shield**:
  Use Shield to protect your infrastructure from DDoS attacks.

- **AWS CloudTrail**:
  Track and log API calls and user activity within your AWS environment, which helps with auditing, security monitoring, and compliance.

Conclusion

Migrating all infrastructure to the Amazon Cloud is definitely feasible, and AWS offers comprehensive services for storage, networking, compute, and security needs. By utilizing the services outlined above, a sysadmin can easily build and manage a complete **Virtual Data Center** in AWS that replaces traditional on-premises infrastructure.

- **Storage**: S3, EBS, EFS
- **Networking**: VPC, Route 53, Direct Connect
- **Servers**: EC2, Lambda
- **Databases**: RDS, Aurora, DynamoDB
- **Security**: IAM, Security Groups, WAF, Shield, CloudTrail

This approach provides flexibility, scalability, and robust security, making it ideal for companies looking to modernize their IT infrastructure and fully embrace cloud computing.