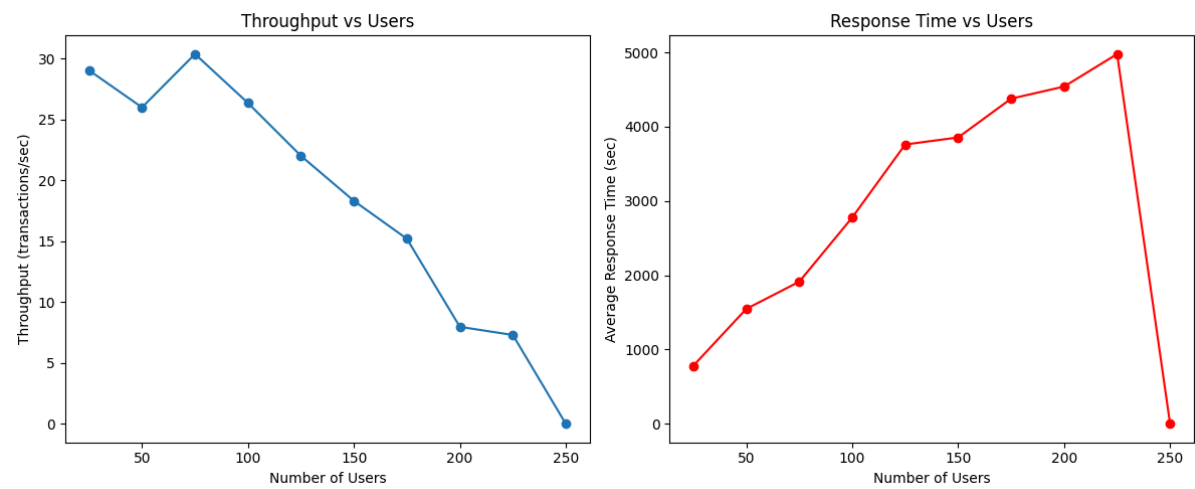


PaaS Auto Scaling

Baseline performance of PaaS



1. From part D, show the inflection point on both of your curves and explain how you map this to the appropriate system resources (performance metrics) to use to trigger auto-scaling for PaaS.

Ans: The inflection point is when the number of concurrent users is 75 users the throughput starts to drop. At that time the CPU Utilization is around 35% so I set the Upper threshold at 30% and lower at 20%

2. Describe the scaling policy you configured to scale up and scale down.

Ans: I configured Scaling cooldown to 60 second, Metric is CPU Utilization since our web app is matrix calculation which use CPU, Statistic to Maximum, Breach duration to 1 Min, Upper threshold 30%, and Lower threshold 20% this make it's very sensitive when our instance CPU Util reach the threshold 30% for 1 minute it's will trigger scaling operation.

| | | |
|----------------------|-----------------------|----------------------|
| Capacity | | |
| Environment type | Min instances | Max instances |
| Load balanced | 1 | 4 |
| Fleet composition | On-demand base | On-demand above base |
| On-Demand instances | 0 | 0 |
| Capacity rebalancing | Scaling cooldown | Processor type |
| Disabled | 60 | x86_64 |
| Instance types | AMI ID | Availability Zones |
| t3.micro | ami-0736ae53e38c421d9 | Any |
| Metric | Statistic | Unit |
| CPUUtilization | Maximum | Percent |
| Period | Breach duration | Upper threshold |
| 1 | 1 | 30 |
| Scale up increment | Lower threshold | Scale down increment |
| 1 | 20 | -1 |

3. How did you run siege (what options) to trigger scaling up and scaling down?

Ans: I ran `siege -c250 -t999H -b http://act-3-php-env.eba-tjaqib2j.us-east-1.elasticbeanstalk.com/`

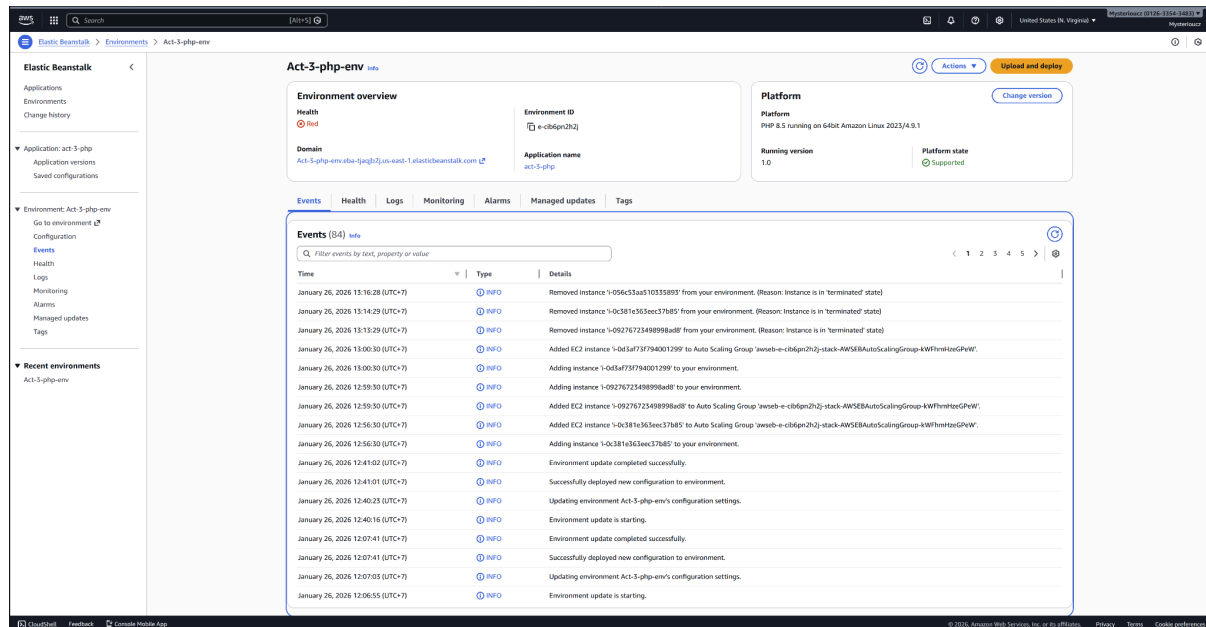
This with 250 concurrent users to allow CPU Utilization to reach above 30% and time for 999 hours or occur until i interrupt execution this is to make sure the CPU utilization will be above 30% for more than 1 minute.

4. Was the scaling up and scaling down behavior consistent with your scaling policy?

Ans: Yes, the scale up and down is consistent with my policy. Scale up when CPU Utilization reaches 30% and Scale down when it is below 20%.

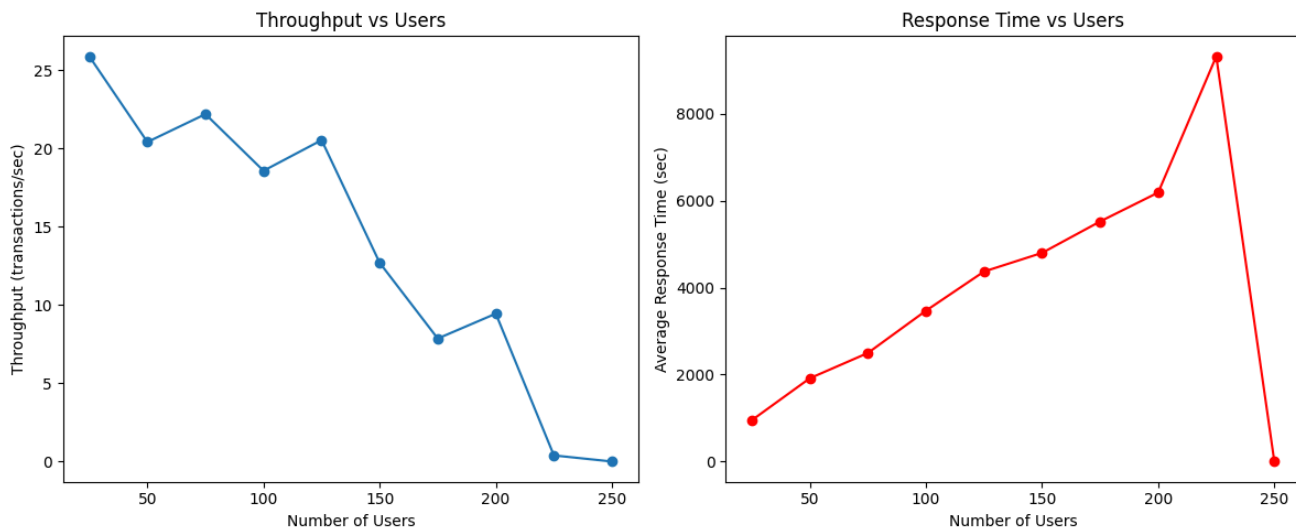
5. Include screenshots from Elastic Beanstalk showing the number of instances running and screenshots of resource monitoring data for your environment in your report to confirm that you successfully triggered auto-scaling according to your configured policies.

Ans: Here is the screen shot of the Event log which shows the ec2 instance added and removed.



IaaS Auto Scaling

Baseline Performance of IaaS



1. Consider observed baseline performance in terms of response time and throughput as a function of offered load (# of client requests/second) when you deploy the same web app to PaaS vs. IaaS. Double-check to see if they are deployed on the same actual instance sizes and check the AZ.

1.1. Discuss the differences in baseline performance in terms of response time and throughput as a function of offered load (# of client requests/second), and conclude which provides better performance?

Ans: From the baseline performance results, PaaS (Elastic Beanstalk) performed better than IaaS. The average response time for PaaS was significantly lower than the manual IaaS setup under the same load conditions.

1.2. Are there any reasons that you think would sensibly explain the performance differences?

Ans: I think it's because PaaS(Elastic Beanstalk) is using Nginx as a reverse proxy and has OPCODE which results in Elastic Beanstalk processes the request and returns the HTML faster than untuned Apache server(IaaS).

2. For IaaS, what scaling policy did you use to scale up and scale down?

Ans:

- Scale Up: Triggered when Maximum CPU Utilization is above 70% for 30 seconds.
- Scale Down: Triggered when Maximum CPU Utilization is below 30% for 30 seconds.

3. How did you run siege (what options) to trigger scaling up and scaling down?

Ans: For scale up I use `siege -c 250 -t 30M -b http://phpiaasgroup-1-407212990.us-east-1.elb.amazonaws.com/`.

This trigger scale up because it increase CPUUtilization to be above 70% for more than 30 seconds and for scale down I use `siege -c 20 -t 3M -v http://phpiaasgroup-1-407212990.us-east-1.elb.amazonaws.com/`.

This trigger scale down because it didn't use that much CPUUtilization(below 30%) for 30 seconds

4. Was the scaling up and scaling down behavior consistent with your scaling policy?

Ans: Yes, the behavior was consistent. The activity logs show new instances launching shortly after the CPU alarm was triggered, and instances being terminated when the load decreased.

5. Include screenshots from EC2 monitoring data in the report to make sure I can see that you did trigger auto-scaling according to your configured policies.

The screenshot shows the AWS Management Console for the 'phpiaasgroup' Auto Scaling Group. The 'Instances' tab is active, displaying a list of three EC2 instances. The first instance, 'i-0da9f163180ea988f', is in a 'Terminated' state. The second instance, 'i-0e9c52443613305a5', is in a 'Running' state. The third instance, 'i-0280cfa548f9a0fa', is also in a 'Running' state. The table includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4 ... , Elastic IP, IPv6 IPs, Monitoring, Security group name, and Key name.

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS | Public IPv4 ... | Elastic IP | IPv6 IPs | Monitoring | Security group name | Key name |
|------|---------------------|----------------|---------------|-------------------|---------------|-------------------|--------------------------|-----------------|------------|----------|------------|---------------------------|----------|
| | i-0da9f163180ea988f | Terminated | t3.micro | | View alarms + | us-east-1a | | | | | disabled | launch-wizard-1,ec2-rds-1 | cu_cloud |
| | i-0e9c52443613305a5 | Running | t3.micro | 3/3 checks passed | View alarms + | us-east-1a | ec2-44-198-158-225.co... | 44.198.158.225 | | | disabled | launch-wizard-1,ec2-rds-1 | cu_cloud |
| | i-0280cfa548f9a0fa | Running | t3.micro | 3/3 checks passed | View alarms + | us-east-1a | ec2-44-202-196-66.co... | 44.202.196.66 | | | disabled | launch-wizard-1,ec2-rds-1 | cu_cloud |

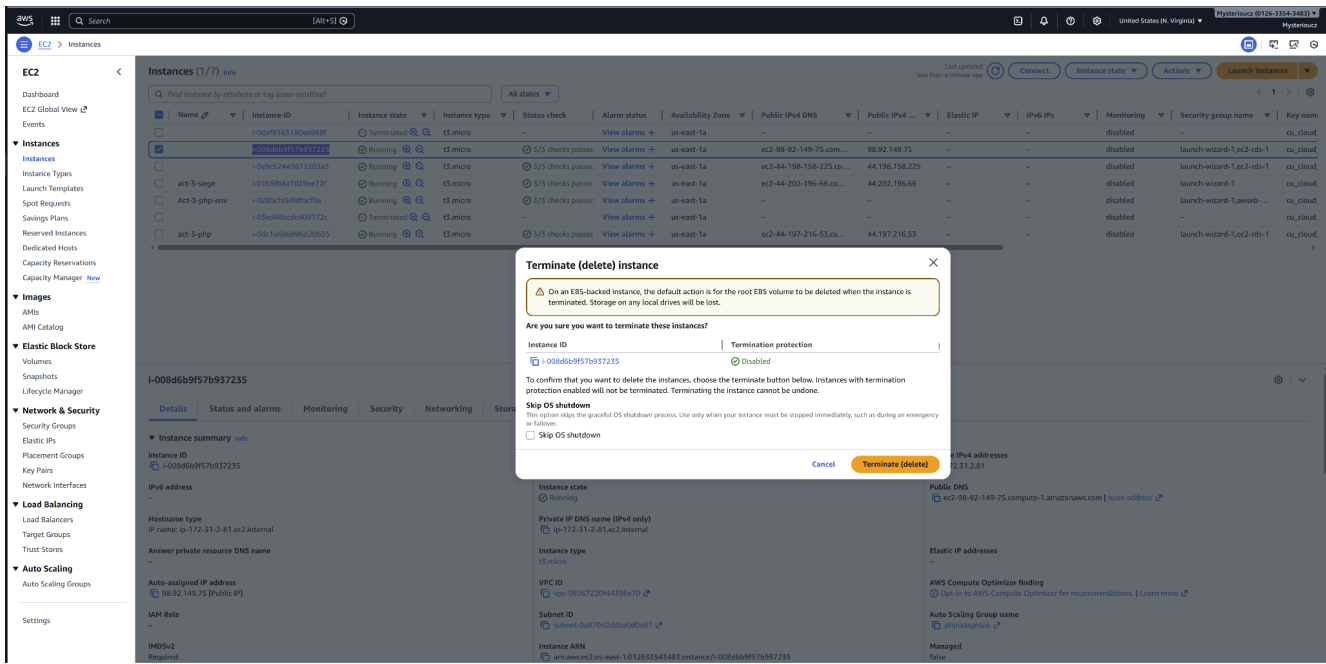
(3 selected instances are managed by AutoScalingGroup named “phpiaasgroup”)

The screenshot shows the AWS Management Console for the 'phpiaasgroup' Auto Scaling Group, specifically the 'Activity logs' section. The logs display a series of successful scaling events triggered by alarms. The events include terminating EC2 instances, launching new EC2 instances, and updating load balancers. The logs are organized into columns for Status, Description, Cause, Start time, and End time.

| Status | Description | Cause | Start time | End time |
|------------|--|--|-------------------------------------|-------------------------------------|
| Successful | Terminating EC2 instance i-05ed48b0c409172c | At 2026-01-26T10:40:54Z a monitor alarm phpiaasgroup scale down alarm in state ALARM triggered policy scale-down changing the desired capacity from 3 to 1. At 2026-01-26T10:41:03Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 3 to 1. At 2026-01-26T10:41:03Z instance i-05ed48b0c409172c was selected for termination. | 2026 January 26, 05:41:03 PM +07:00 | 2026 January 26, 05:46:46 PM +07:00 |
| Successful | Terminating EC2 instance i-0da9f163180ea988f | At 2026-01-26T10:35:54Z a monitor alarm phpiaasgroup scale down alarm in state ALARM triggered policy scale-down changing the desired capacity from 3 to 2. At 2026-01-26T10:35:55Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 3 to 2. At 2026-01-26T10:36:00Z instance i-0da9f163180ea988f was selected for termination. | 2026 January 26, 05:36:00 PM +07:00 | 2026 January 26, 05:42:23 PM +07:00 |
| Successful | Launching a new EC2 instance i-0e9c52443613305a5 | At 2026-01-26T10:30:58Z a monitor alarm phpiaasgroup scale up alarm in state ALARM triggered policy scale-up changing the desired capacity from 2 to 3. At 2026-01-26T10:31:06Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 3. | 2026 January 26, 05:31:08 PM +07:00 | 2026 January 26, 05:31:13 PM +07:00 |
| Successful | Launching a new EC2 instance i-0da9f163180ea988f | At 2026-01-26T10:25:58Z a monitor alarm phpiaasgroup scale up alarm in state ALARM triggered policy scale-up changing the desired capacity from 1 to 2. At 2026-01-26T10:26:02Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2. | 2026 January 26, 05:26:04 PM +07:00 | 2026 January 26, 05:26:11 PM +07:00 |
| Successful | Updating load balancers/target groups: Successful. Status Reason: Added: amazon-elasticloadbalancing-us-east-1:1072635354383:target-group/phpiaasgroup-1/9dcd1da77264b492 (Target Group) | | 2026 January 26, 03:46:46 PM +07:00 | 2026 January 26, 03:46:46 PM +07:00 |
| Successful | Launching a new EC2 instance i-05ed48b0c409172c | At 2026-01-26T08:46:32Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 1. At 2026-01-26T08:46:32Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1. | 2026 January 26, 03:46:37 PM +07:00 | 2026 January 26, 03:46:43 PM +07:00 |

(Activity logs of Auto Scaling group “phpiaasgroup”)

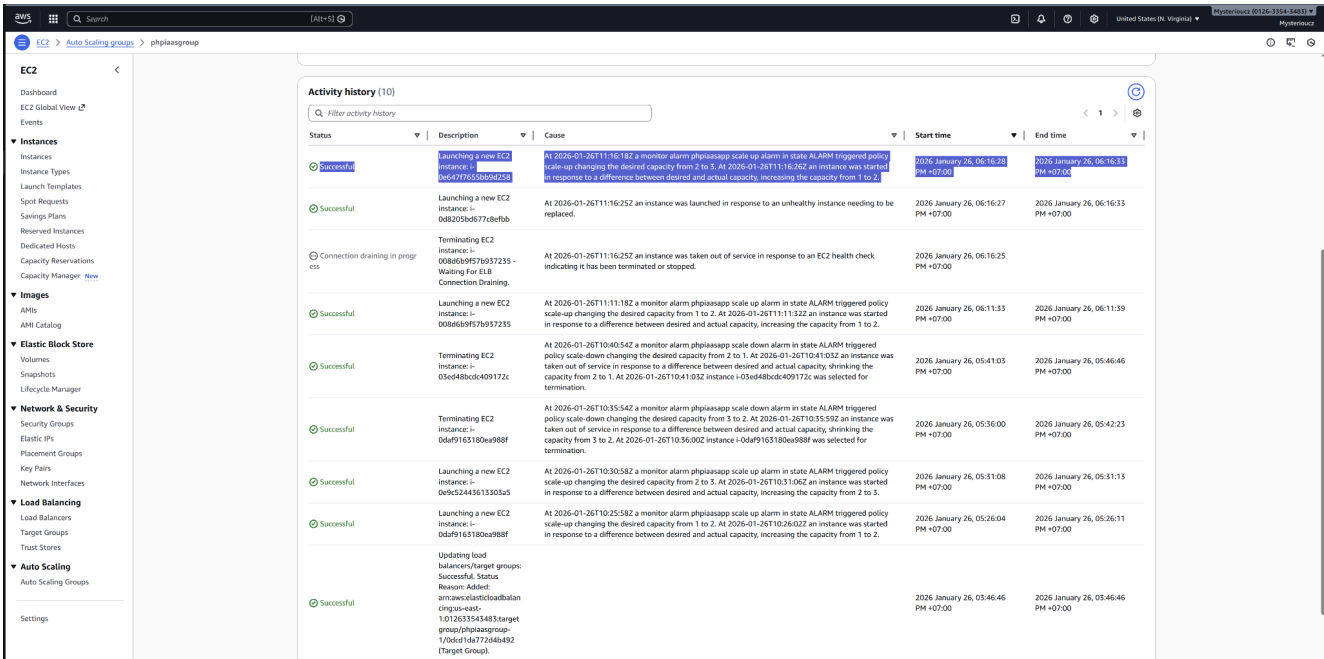
6. Consider what Amazon did when you terminated your EC2 instance(s) in part D (fault tolerance).



(terminated an instance)

6.1. Is Amazon able to launch new instances to replace the one(s) you killed?

Ans: Yes. The Auto Scaling Group detected the unhealthy (terminated) instance and automatically launched a new replacement instance to maintain the desired capacity.



(Activity history that Auto Scaling group launching new instances 2 upper rows)

EC2

Dashboard

EC2 Global View

Events

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Capacity Manager

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

Load Balancing

Load Balancers

Target Groups

Trust Stores

Auto Scaling

Auto Scaling Groups

Settings

Instances (1/9)

Find Instance by attribute or tag (case-sensitive)

All states

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS | Public IPv4 ... | Elastic IP | IPv6 IPs | Monitoring | Security group name | Key nam |
|------|---------------------|----------------|---------------|-------------------|---------------|-------------------|--------------------------|-----------------|------------|----------|------------|---------------------------|----------|
| | i-0da9163180ea988f | Terminated | t5.micro | | View alarms + | us-east-1a | - | - | - | - | disabled | - | cu_cloud |
| | i-008d6a9f570937235 | Terminated | t5.micro | | View alarms + | us-east-1a | - | - | - | - | disabled | - | cu_cloud |
| | i-0a3c52443611303a5 | Running | t5.micro | 3/3 checks passed | View alarms + | us-east-1a | ec2-44-198-158-225.co... | 44.198.158.225 | - | - | disabled | launch-wizard-1-ec2-rds-1 | cu_cloud |
| | i-0d8205bd677c8efbb | Running | t5.micro | Initializing | View alarms + | us-east-1a | ec2-44-192-37-92.com... | 44.192.37.92 | - | - | disabled | launch-wizard-1-ec2-rds-1 | cu_cloud |
| | i-0280c7d548facc9a | Running | t5.micro | 3/3 checks passed | View alarms + | us-east-1a | ec2-44-202-196-66.co... | 44.202.196.66 | - | - | disabled | launch-wizard-1 | cu_cloud |
| | i-03e848bcd409172c | Terminated | t5.micro | | View alarms + | us-east-1a | - | - | - | - | disabled | launch-wizard-1-awsse- | cu_cloud |
| | i-0dc1a6b6896d20b35 | Running | t5.micro | 3/3 checks passed | View alarms + | us-east-1a | ec2-44-197-216-53.co... | 44.197.216.53 | - | - | disabled | launch-wizard-1-ec2-rds-1 | cu_cloud |
| | i-0e6477f53bba9d258 | Running | t5.micro | Initializing | View alarms + | us-east-1a | ec2-18-215-187-93.co... | 18.215.187.93 | - | - | disabled | launch-wizard-1-ec2-rds-1 | cu_cloud |

i-0d8205bd677c8efbb

Details

Status and alarms

Monitoring

Security

Networking

Storage

Tags

Instance summary

Instance ID

IP v6 address

Hostname type

Answer private resource DNS name

Auto-assigned IP address

IAM Role

IMDSv2

Public IPv4 address

Instance state

Private IP DNS name (IPv4 only)

Instance type

VPC ID

Subnet ID

Instance ARN

Private IPv4 addresses

Public DNS

Elastic IP addresses

AWS Compute Optimizer finding

Auto Scaling Group name

Managed

(The lowest and selected row is the latest instances that got launched)

6.2. When the new instances are launched, are your web clients (browser or siege) able to access the web application like normal? Provide evidence of what your web clients see.

Ans: Yes. Once the new instance passed its health checks, the Load Balancer resumed sending traffic to it. My Siege client and browser could access the web application normally without manual intervention.

Congratulations!

Your AWS Elastic Beanstalk PHP application is now running on your own dedicated environment in the AWS Cloud

You are running PHP version 8.5.1

Matrix multiplication completed.

Cloud Computing Auto Scaling

Engineered for Heavy Load: 43350

What's Next?

- AWS Elastic Beanstalk overview
- Deploying AWS Elastic Beanstalk Applications in PHP Using CLI and GUI
- Using Amazon RDS with PHP
- Customizing the Software on EC2 Instances
- Customizing Environment Resources

AWS SDK for PHP

- AWS SDK for PHP home
- PHP developer center
- AWS SDK for PHP on GitHub

```
ec2-user@ip-172-31-15-173:~$ curl -v http://172.31.15.173/
* Host 172.31.15.173:80 was successfully cached (vs. 0).
* Connected to 172.31.15.173 (172.31.15.173) port 80
* GET http://172.31.15.173/ HTTP/1.1
* 0 1438 bytes => GET /
* 0.09 secs: 1438 bytes => GET /
* 0.02 secs: 193 bytes => GET /css?family=Lobster+Two
* 0.10 secs: 1438 bytes => GET /
* 0.19 secs: 1438 bytes => GET /
* 0.10 secs: 3027 bytes => GET /
* 0.08 secs: 3490 bytes => GET /styles.css
* 0.14 secs: 3027 bytes => GET /
* 0.02 secs: 193 bytes => GET /css?family=Lobster+Two
* 0.01 secs: 3490 bytes => GET /styles.css
* 0.10 secs: 3027 bytes => GET /
* 0.01 secs: 3490 bytes => GET /styles.css
* 0.03 secs: 193 bytes => GET /css?family=Lobster+Two
* 0.02 secs: 193 bytes => GET /css?family=Lobster+Two
* 0.12 secs: 3027 bytes => GET /
* 0.00 secs: 3490 bytes => GET /styles.css
* 0.02 secs: 193 bytes => GET /css?family=Lobster+Two
* 0.11 secs: 1438 bytes => GET /
* 0.10 secs: 3027 bytes => GET /
```

7. Discuss the differences between configuring auto-scaling in IaaS vs. auto-scaling PaaS. What are the strengths of each configuration approach?

Ans: Both can achieve the same configuration but there are some differences.

- **PaaS (Elastic Beanstalk):** Configuration is much simpler. All settings (scaling triggers, load balancing, health checks) are centralized on a single "Configuration" dashboard.
 - **IaaS (Manual):** Requires more manual steps. I had to separately configure the Launch Template, Target Groups, Load Balancer, and Auto Scaling Group policies.
-

8. Discuss the differences between IaaS vs. PaaS. What are the strengths of each approach?

Ans: I think both have trade offs.

- **IaaS Strengths:** Offers higher flexibility and control. We can customize the OS, install specific kernel modules, and tune the network stack exactly how we need it.
 - **PaaS Strengths:** Offers higher convenience and speed. It handles deployment, patching, and scaling logic automatically, allowing developers to focus on code rather than infrastructure.
-