Amazon Web Services provides many services to any size of enterprise. Many organizations have started to shift their data centers to the cloud because of the many benefits it has. Many organizations are using AWS EC2 service to deploy their virtual servers, web applications and many more. There are many types of EC2 instances for different applications. New Amazon Web Services has more than one millions users of their service and it is getting higher and higher. In the present scenario, users select the best EC2 type that is compatible with most of the incoming traffic. However, this approach is not efficient while considering performance, resource and price. It has been observed that while meeting the performance criteria, resources are wasted in a huge amount that eventually leads to higher cost to the organization. To make it cost efficient, devops engineers have to manually upgrade or downgrade their cloud instances. It is next to impossible for engineers to constantly monitor resource utilization of their instances and make changes to their instances.

Hence, we are proposing the automation technique for scaling EC2 instances while meeting quality of service and cost efficiency. The idea of vertical scale is to scale up or scale down EC2 instances while meeting quality of service and price. It has been observed that while meeting the performance criteria, resources are wasted in a huge amount that eventually leads to higher cost to the organization. To make it cost efficient, devops engineers have to manually upgrade or downgrade their cloud instances. It is next to impossible for engineers to constantly monitor resource utilization of their instances and make changes to their instances.

### Methodology

#### Algorithm to find optimal instance

The algorithm takes variables as the arguments that include Instance type, CPU utilization and Memory of running instance. The algorithm starts running if CPU utilization of the running instance crosses the upper and lower boundary defined by admin. Next step is to find the value of Required Memory that belongs to the predefined boundary. This can be done by dividing current memory value with boundary values.

For example, an instance with 1024 MB RAM is running with 80 % of CPU utilization which is greater than upper boundary. That means 820 MB RAM is currently used and new required memory RAM is 820/75% and 820/25% i.e. Required RAM should be between 1092 and 3277 MB. The algorithm knows that the new instance should have memory that is between above mentioned values. In the next step, the algorithm goes through the database and fetches all the instances that have memory in the above range. The algorithm uses NoSQL database that contains the information of all the 200 EC2 instance types. Moreover, filtered instances should belong to the same category. The algorithm knows the pricing of each instance that is being filtered in the above step. In the final step, Algorithm returns the instance type that has the lowest price.

#### Analysis and Results

Nowadays, Amazon web services are providing only Horizontal scaling in their operation of auto-scaling. Horizontal Scaling means when the load on instance hikes instantly then Auto scaling feature increases the number of instances and manages the load on the server. So the price of the instance is getting multiple with several instances.

For example a company is using m5dn.xlarge instance and when load on that server increases then auto-scaler double this instance and price for these two instances will be $198.56 per month. With Ec2 auto transition script will decide the best instance for increasing traffic. It is going to choose m5a.2xlarge and price for this $251.12 per month. If we use auto-scaler then our cost is $397.04.

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>RAM (MB)</th>
<th>CPU Utilization</th>
<th>Price per Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>M5D Extra Large</td>
<td>820</td>
<td>80%</td>
<td>$397.04</td>
</tr>
<tr>
<td>M5A 2XLarge</td>
<td>1092</td>
<td>75%</td>
<td>$251.12</td>
</tr>
</tbody>
</table>

To automate vertical scaling, we designed a new architecture and a technique to scale our EC2 instance according to incoming traffic. Continued monitoring is done on the current instance to measure CPU utilization, memory usage etc. To monitor real time usage, we have used AWS cloudwatch. According to policy deployed by the user, a smart decision is taken for selecting new instance types and it is deployed into the production environment. For smart decision taking, we used AWS Lambda service. Bootstrapping is also significant while launching the instances. To install "Golden Image", bootstrap code deployed using AMI with all required configurations.

### Summary/Conclusions

In a nutshell, our product is beneficial to many companies to reduce their cost on web service and companies can get the best EC2 instance for their company as their need. In addition it is getting scale up or scale down with zero down time. Auto Transition algorithm chooses the most cost efficient EC2 instance for the company. It deletes the old instance and copies AMI to the new cost efficient instance EC2 instance. CPU utilization of EC2 is always between 25% - 75%. Cloudwatch monitor utilization of EC2 instance and generated SNS query and triggered lambda code. In the lambda function check the NoSQL database and select the cheap instance for lower requirements.

In the future, we can implement Machine Learning. Time based scaling. Consider RAM, throughput for scaling purpose. With that we can make a more reliable and robust auto-scaler. Well trained machine learning models are used to predict the traffic on the running application. The model needs at least 24 hours of data to trigger the functionality and can do a prediction for the next 45 hours. Furthermore vertical scaling can be implemented on GCP and Azure. Our algorithm is based on only CPU utilization but in future that could be on all elements like network in packets, network out packets, bandwidth, input and output operation of the EC2 instance. And developers can combine all EC2 instance types to get the best result.