Analyzing performance and issues resolution in auto scaling using logs

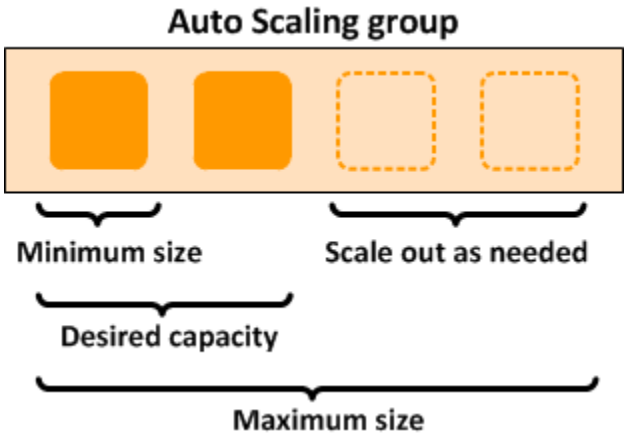
Outline:

1. Introduction to AWS auto scaling
2. Introduction to auto scaling architecture
3. Components of AWS auto scaling
4. Different types of auto scaling log events
    i. EC2 Instance-launch Lifecycle Action
    ii. EC2 Instance Launch Successful
    iii. EC2 Instance Launch Unsuccessful
    iv. EC2 Instance-terminate Lifecycle Action
    v. EC2 Instance Terminate Successful
    vi. EC2 Instance Terminate Unsuccessful
5. Business Problems in auto scaling use
6. Analytics of performance related data
7. Benefits of Auto Scaling
8. Analyzing Cost using current configuration
9. How do we come up with optimal cost
10. Problems one can encounter in auto scaling
11. use cases of auto scaling architecture in digiverifier
12. Effective usage of Auto Scaling functionality

1. Introduction to AWS Auto Scaling

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application.

Auto scaling group will have collection of instances and you can specify minimum , maximum and desired number of instance. Instances will never go down below minimum size.



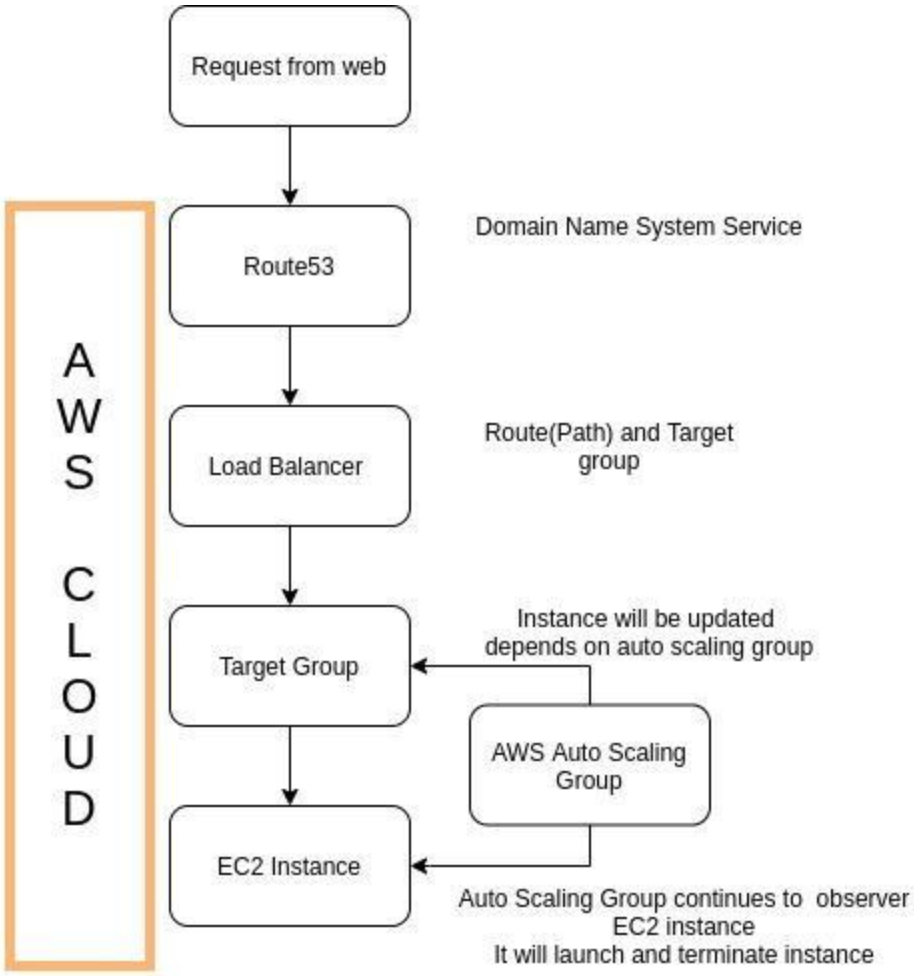
Auto scaling Group continues to launch and terminate instance. There is a conflict In executing scale-in and scale-out policy. To understand performance related issues we must dig deeper in to the log events of auto scaling.

2. Introduction to Auto Scaling Architecture

The first step in auto scaling is the creation of the **launch configuration**. And the second part is the creation of an **auto scaling group**. These two define how an auto scaling group builds new EC2 instances. Auto Scaling group continues to observe EC2 instance CPU memory utilization and take action as configured.

Auto Scaling provides a simple, powerful user interface that lets you build scaling plans for Amazon EC2 instances.We can create scale in and scale out policy as per our requirement which will create instance and terminate instance.

3. Components of AWS Auto Scaling



Below are the AWS auto scaling components

- Auto Scaling launch configuration
- Auto Scaling Group
- EC2 instance
- Security Group
- Amazon Machine image (AMI)
- Elastic Load Balancer (ELB)
- Virtual Private Cloud(VPC)

Auto Scaling launch configuration: In launch configuration we will select AMI and type of EC2 instance. Mainly you specify how your newly launch instance should be and what services to run in that instance using user scripts.

Auto Scaling Group: Specify your scale in and Scale out policy. We have to set all the network related configurations like VPC and security group.

4. Different types of Auto Scaling log events

The following are the different types of log events in auto Scaling:

1. EC2 Instance-launch Lifecycle Action
2. EC2 Instance Launch Successful
3. EC2 Instance Launch Unsuccessful
4. EC2 Instance-terminate Lifecycle Action
5. EC2 Instance Terminate Successful
6. EC2 Instance Terminate Unsuccessful

Basic information that can be extracted from these Log events.

a) Detail-type(Eventype/action)

b) Source

c) Resource ARN

d) EC2 instance-id

e) Instance-id(launched/terminated)

f) Cause of action

g) AutoScalingGroupName

I. EC2 Instance-launch Lifecycle Action

The following sample event occurs when instance launched and went to wait state due to life cycle actions done on the newly created instance.

```
{
 "version": "0",
 "id": "12345678-1234-1234-1234-123456789012",
 "detail-type": "EC2 Instance-launch Lifecycle Action",
 "source": "aws.autoscaling",
 "account": "123456789012",
 "time": "yyyy-mm-ddThh:mm:ssZ",
 "region": "us-west-2",
 "resources": [
  "auto-scaling-group-arn"
 ],
 "detail": {
  "LifecycleActionToken": "87654321-4321-4321-4321-210987654321",
  "AutoScalingGroupName": "my-asg",
  "LifecycleHookName": "my-lifecycle-hook",
  "EC2InstanceId": "i-1234567890abcdef0",
  "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
  "NotificationMetadata": "additional-info"
 }
}
```

i. **EC2 Instance Launch Successful**

The following sample event occurs when instance is launched successful because of auto scaling.

```
{
 "version": "0",
 "id": "12345678-1234-1234-1234-123456789012",
 "detail-type": "EC2 Instance Launch Successful",
 "source": "aws.autoscaling",
 "account": "123456789012",
 "time": "yyyy-mm-ddThh:mm:ssZ",
 "region": "us-west-2",
 "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
 ],
 "detail": {
    "StatusCode": "InProgress",
    "Description": "Launching a new EC2 instance: i-12345678",
    "AutoScalingGroupName": "my-auto-scaling-group",
```

```
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
       "Availability Zone": "us-west-2b",
       "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text"
  }
}
```

ii. EC2 Instance Launch Unsuccessful

The following sample event occurs when instance is launched but unsuccessful because of auto scaling.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Launch Unsuccessful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "Failed",
    "AutoScalingGroupName": "my-auto-scaling-group",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
       "Availability Zone": "us-west-2b",
       "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "message-text",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
```

```
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text"
  }
}
```

iii. EC2 Instance-terminate Lifecycle Action

The following sample event occurs when instance terminated and went to wait state due to life cycle actions done on the newly terminated instance.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken":"87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName":"my-asg",
    "LifecycleHookName":"my-lifecycle-hook",
    "EC2InstanceId":"i-1234567890abcdef0",
    "LifecycleTransition":"autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata":"additional-info"
  }
}
```

iv. EC2 Instance Terminate Successful

The following sample event occurs when instance is terminated successful because of auto scaling.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Terminate Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
```

```
 "resources": [
  "auto-scaling-group-arn",
  "instance-arn"
 ],
 "detail": {
    "StatusCode": "InProgress",
    "Description": "Terminating EC2 instance: i-12345678",
    "AutoScalingGroupName": "my-auto-scaling-group",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
       "Availability Zone": "us-west-2b",
       "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text"
 }
}
```

v. **EC2 Instance Terminate Unsuccessful**

The following sample event occurs when instance is terminated but unsuccessful because of auto scaling.

```
{
 "version": "0",
 "id": "12345678-1234-1234-1234-123456789012",
 "detail-type": "EC2 Instance Terminate Unsuccessful",
 "source": "aws.autoscaling",
 "account": "123456789012",
 "time": "yyyy-mm-ddThh:mm:ssZ",
 "region": "us-west-2",
 "resources": [
  "auto-scaling-group-arn",
  "instance-arn"
 ],
 "detail": {
    "StatusCode": "Failed",
    "AutoScalingGroupName": "my-auto-scaling-group",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
       "Availability Zone": "us-west-2b",
```

```
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "message-text",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text"
  }
}
```


What is Lifecycle hook

Lifecycle hooks enable us to perform custom actions by pausing instances. when Auto Scaling group launches or terminates them. When an instance is paused, it remains in a wait state either until you complete the lifecycle action or time out.

Lifecycle action can be performed using command or operation or until the time out period ends. We call this time out period as Heartbeat timeout, by default it is one hour.

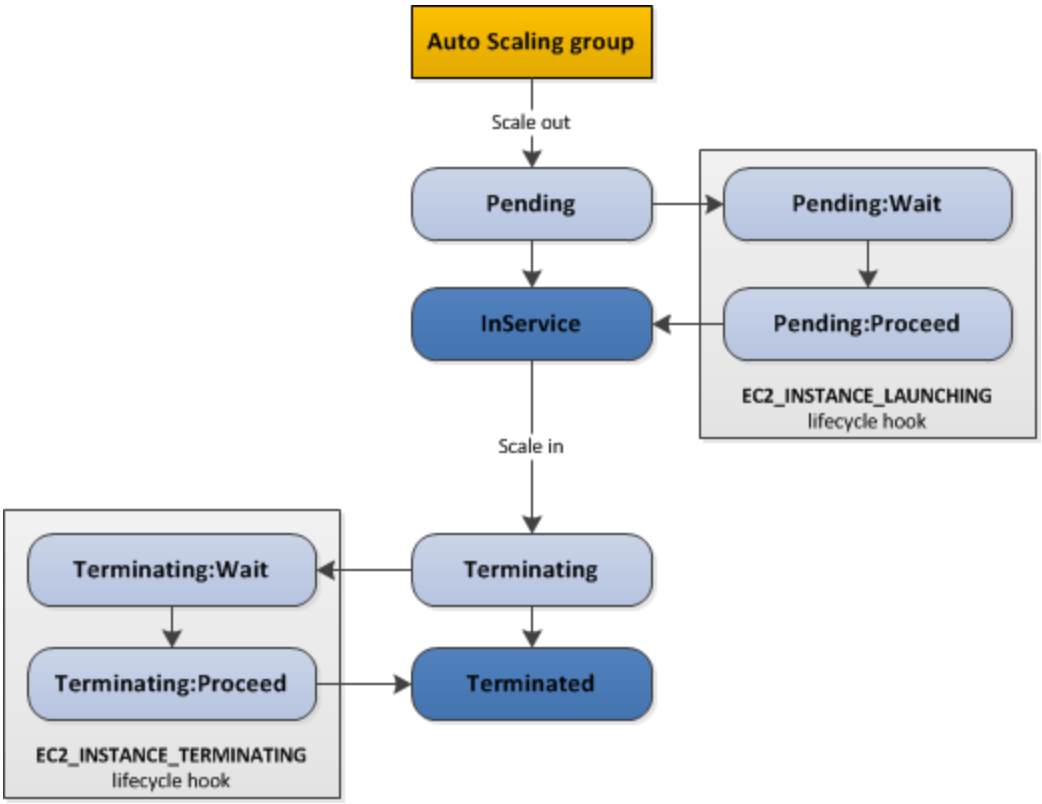
Example for Launch instance:

Before auto scaling launches newly created instance lifecycle hook will move that instance in to waiting state.While the instance is in a wait state, you can install or configure software on it, making sure that your instance is fully ready before it starts receiving traffic.

Example for terminate instance:

When auto scaling terminates instance first it will deregister from load balancer if it is configured. Now lifecycle hook will pass the instance when it is in wait state we can connect to the instance and download logs or other date before it completely terminated.

Life cycle hook Flowchart



How to configure Life cycle hook

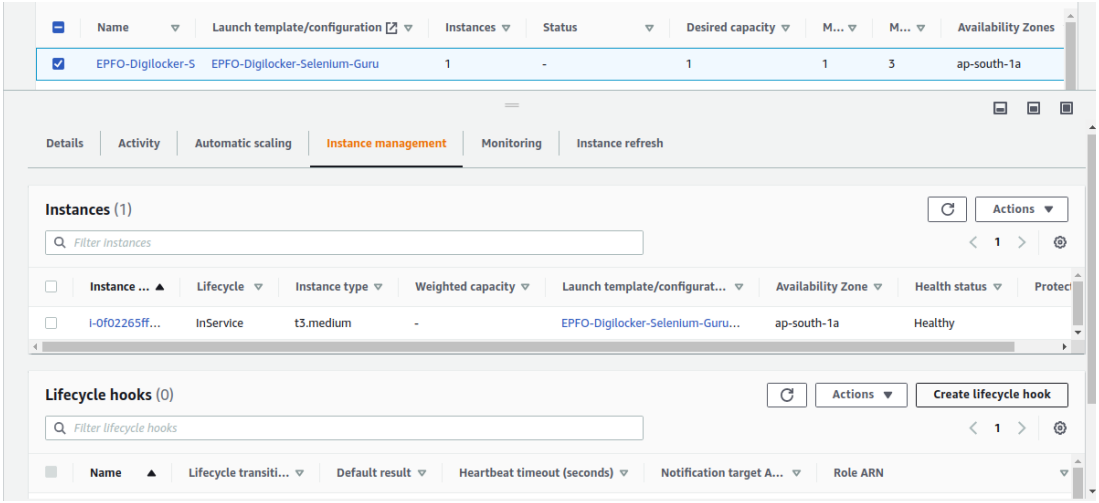
Lifecycle hooks can be configured only when instance is lunching and terminating by using auto scaling. Each Auto Scaling group can have multiple lifecycle hooks. However, there is a limit on the number of hooks per Auto Scaling group.

- Lifecycle hooks per Auto Scaling group: 50
- You can add **Heartbeat timeout** value. The value must be from 30 to 7200 seconds.

Select Auto scaling Group and click on Instance management tab, in Lifecycle hooks, choose Create lifecycle hook. After you add lifecycle hooks to your Auto Scaling group, they work as follows:

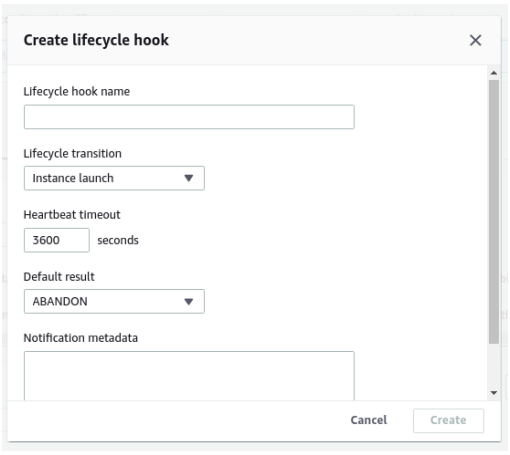
1. The Auto Scaling group responds to scale-out events by launching instances and scale-in events by terminating instances.
2. The lifecycle hook puts the instance into a wait state. The instance is paused until you continue or the timeout period ends.
3. You can perform a custom action

4. By default, the instance remains in a wait state for one hour, and then the Auto Scaling group continues the launch or terminate process. If you need more time, you can restart the timeout period by recording a heartbeat. If you finish before the timeout period ends, you can complete the lifecycle action, which continues the launch or termination process.



Following is the creation of lifecycle hook

- We can set the Heartbeat time and default is one hour
- We need to specify lifecycle transition whether it is for launching or termination

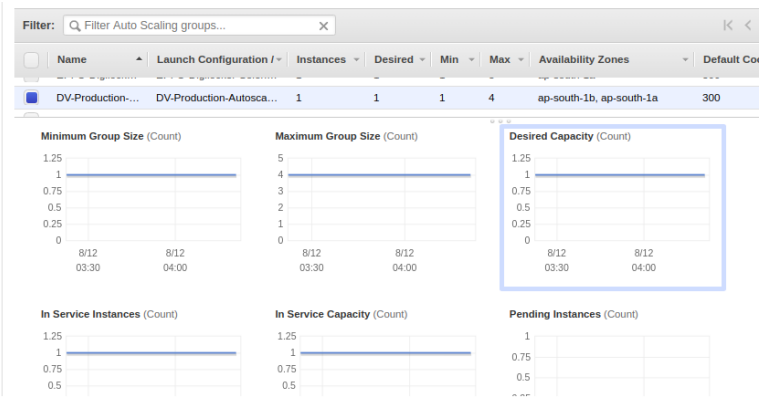


5. Business Problems in using auto scaling

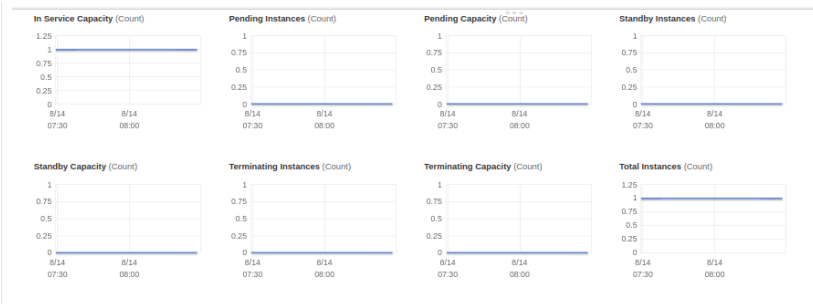
- Some times EC2 Auto Scaling group isn’t executing the scaling policy what we configured to do.
- EC2 Auto Scaling group is executing scaling out action instead of scaling in action.
- Amazon EC2 Auto Scaling group continuously launches and terminates Amazon EC2 instances.
- EC2 Instance is neither launched nor terminated but got stuck in the Pending:Wait or Terminating:Wait state during scaling activity.
- EC2 Auto Scaling group isn’t responding to a scheduled action configured.

6. Analytics of performance related data

Performance of AWS auto scaling can be seen in dashboards as below.Dashboard will have graphical representation of configured numbers such as minimum, maximum and desired number of instance.



Also we can see how many instances are active now and how many are in pending state.



7. Benefits of Auto Scaling

- AWS Auto Scaling makes scaling simple with recommendations that allow you to optimize performance, cost.
- It helps you to monitor your utilization and cost efficiencies while using the services of AWS. This helps to pay only for what you have utilized and what you need. AWS manages the capacity used and notifies the user according to it. AWS Autoscaling is free and removes the quantity, not in use and thus, helps to avoid overspending.
- Better availability. Amazon EC2 Auto Scaling helps ensure that your application always has the right amount of capacity to handle the current traffic demand.

8. Analyzing the cost using current configuration

9. How do we come up with optimal cost

When we launch a new instance, we need to provide the basic configurations like RAM, CPU cores, memory etc. But this configuration can't be changed later, whenever scale-out happens. To meet the demand definitely need to launch a new instance with higher configurations. But it is useless when there is no load on instance.

Since large number of instances available listing out most commonly used ones.

Instance type	vCPU	Instance Storage (GB)	Linux/UNIX Usage
t3.small	2	2 GiB EBS Only	$0.0224 per Hour
t3.medium	2	4 GiB EBS Only	$0.0448 per Hour
t3.large	2	8 GiB EBS Only	$0.0896 per Hour
t3.xlarge	4	16 GiB EBS Only	$0.1792 per Hour

Ref: https://aws.amazon.com/ec2/pricing/on-demand/

10. Problems one can encounter in auto scaling:

- Some times EC2 Auto Scaling group isn't executing the scaling policy what we configured to do.
- EC2 Auto Scaling group is executing scaling out action instead of scaling in action.

- Amazon EC2 Auto Scaling group continuously launches and terminates Amazon EC2 instances.
- EC2 Instance is neither launched nor terminated but got stuck in the Pending: Wait or Terminating: Wait state during scaling activity.
- EC2 Auto Scaling group isn't responding to a scheduled action configured.

11. Use case of auto scaling architecture in digiverifier:

Initially digiverifier was using m4 large(8gb) to run the application, continues to use for a year and realized network traffic and CPU load usage is very less which can be handled by an instance with smaller RAM size instance. With the help of auto scaling digiverifier reduces the instance from m4 large to t3 medium, if there is scale out happens, auto scaling feature will automatically launch a new instance and meet the necessary requirements. By this way digiverifier investment on ec2 instance has reduced to 58%.

12. Benefits & Effective usage of auto scaling

1. Setup Scaling Quickly

AWS Auto Scaling enables us to set target utilization levels for multiple resources in a single, intuitive interface. We can quickly see the average utilization of all of your scalable resources without having to navigate to other consoles. For example, if an application uses Amazon EC2 and Amazon DynamoDB, we can use AWS Auto Scaling to manage resource provisioning for all of the EC2 Auto Scaling groups and database tables in your application

2. Automatically Maintain Performance:

Using AWS Auto Scaling, you maintain optimal application performance and availability, even when workloads are periodic, unpredictable, or continuously changing. AWS Auto Scaling continually monitors your applications to make sure that they are operating at your desired performance levels. When demand spikes, AWS Auto Scaling automatically increases the capacity of constrained resources so you maintain a high quality of service.

3. Pay as You use:

AWS Auto Scaling can help you optimize your utilization and cost efficiencies when consuming AWS services so you only pay for the resources you actually need. When demand drops, AWS Auto Scaling will automatically remove any excess resource capacity so you avoid overspending. AWS Auto Scaling is free to use, and allows you to optimize the costs of your AWS environment.

Ref: https://aws.amazon.com/autoscaling/

Conclusion

Using Auto scaling log events we can solve problems and monitor auto scaling groups. Also we will get more insights to be proactive.