Connection draining

Time to complete “in-flight-requests” while the instance is de-registering or un-healthy

ASG:

Scale out🡺add EC2 instances-> to match an increased load

Scale in->remoce Ec2 instances🡪to match a decreased load

Ensure we have a minimum and maximum number of EC2 instances running

Automatically register new instances to a load balancer

Re-create an EC2 instance in case previous one is terminated (ex:un-healthy)

\*\* ASG are free (you only pay for underlying Ec2 instances)

Minimum capacity: instances required minimum for ASG

Desired Capacity: how many instances you want in the ASG

Maximum capacity: scale-out instances as needed

ASG Scaling group Attributes:

A launch template

-🡪how to launch instances with in an ASG

AMI+ Instance Type

Ec2 User data

ESB Volumes

SG

SSH Key pair

IAM Role for Ec2 instance

Network and subnet information

Load Balancer information

Min/max size capacity

Scaling policies

Possible to scale-in and scale-out based oncloud watch alaram

Metrics such as average cpu are computed for all overall asg instances

Target Tracking scaling

Want the asg cpu to stay at around 40%

Simple/Step Scaling:

When cloud watch alarm is triggered (cpu>70%) then add 2 units

When cloud watch alarm is triggered (cpu<30%) then remove 1 units

Scheduled Actions:

Anticipate scaling based on know usage pattern

Increase the min capacity to 10 at 5 pm on Fridays

Predictive scaling☹more advanced way of scaling based on machine learning

Based on forecast analysis instances are scaled out and scaled in.

Good Metrics to scale on

Cpu utilization:Average CPU utilization across your instances

RequestCountPerTarget:to make sure number of requests per EC2 instance is stable

Average Network In/Out:some uploads and downloads reaches threshold in Ec2 instances based on which we can control ASG

Any Custom metric:that we can push using CloudWatch

sudo amazon-linux-extras install epel -y

sudo yum install stress -y

stress -c 4