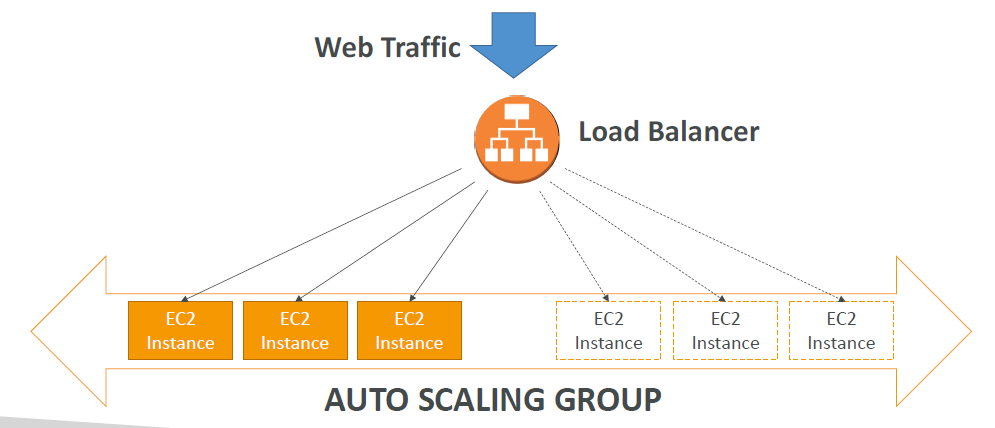
**Auto Scaling Group [ASG]**

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. You create collections of EC2 instances, called Auto Scaling groups. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes above this size.

* The more users the more load you might have and vise versa.
* The goal of an Auto Scaling Group [ASG] is to :
* Scale out [add EC2 instances] to match an increased load.
* Scale in [remove EC2 instances] to match a decreased load.
* Ensure we have a minimum and a maximum number of machines running
* Automcatically register new instances to a load balancer.

****

**Attributes:**

• A launch configuration

• AMI + Instance Type

• EC2 User Data

• EBS Volumes

• Security Groups

• SSH Key Pair

• Min Size / Max Size / Initial Capacity

• Network + Subnets Information

• Load Balancer Information

• Scaling Policies

**Auto Scaling Alarms**

It is possible to scale an ASG based on **CloudWatch alarms**

• An Alarm monitors a metric (such as Average CPU)

• Metrics are computed for the overall ASG instances[avg]

• Based on the alarm:

• We can create scale-out policies (increase the number of instances)

• We can create scale-in policies (decrease the number of instances)

**Auto Scaling New Rules**

* + It is now possible to define ”better” auto scaling rules that are directly

managed by EC2

* + Target Average CPU Usage
  + Number of requests on the ELB per instance
  + Average Network In
  + Average Network Out
  + These rules are easier to set up and can make more sense

**Auto Scaling Custom Metric**

* + We can auto scale based on a custom metric (ex: number of connected

users)

* + 1. Send custom metric from application on EC2 to CloudWatch

(PutMetric API)

* + 2. Create CloudWatch alarm to react to low / high values
  + 3. Use the CloudWatch alarm as the scaling policy for ASG

**ASG Brain Dump**

* Scaling policies can be on CPU, Network… and can even be on custom metrics or

based on a schedule (if you know your visitors patterns)

* ASGs use Launch configurations or Launch Templates (newer)
* To update an ASG, you must provide a new launch configuration / launch template
* IAM roles attached to an ASG will get assigned to EC2 instances
* ASG are free. You pay for the underlying resources being launched
* Having instances under an ASG means that if they get terminated for whatever reason, the ASG will automatically create new ones as a replacement. Extra safety!
* ASG can terminate instances marked as unhealthy by an LB (and hence replace them)

**Auto Scaling Groups – Scaling Policies**

**Target Tracking Scaling**

* Most simple and easy to set-up
* Example: I want the average ASG CPU to stay at around 40%

**Simple / Step Scaling**

* When a CloudWatch alarm is triggered (example CPU > 70%), then add 2 units
* When a CloudWatch alarm is triggered (example CPU < 30%), then remove 1

**Scheduled Actions**

* Anticipate a scaling based on known usage patterns
* Example: increase the min capacity to 10 at 5 pm on Fridays

**Auto Scaling Groups - Scaling Cooldowns**

* The cooldown period helps to ensure that your Auto Scaling group doesn't

launch or terminate additional instances before the previous scaling activity takes effect.

* In addition to default cooldown for Auto Scaling group, we can create cooldowns that apply to a specific simple scaling policy
* A scaling-specific cooldown period overrides the default cooldown period.
* One common use for scaling-specific cooldowns is with a scale-in policy—a policy that terminates instances based on a specific criteria or metric. Because this policy terminates instances, Amazon EC2 Auto Scaling needs less time to determine whether to terminate additional instances.
* If the default cooldown period of 300 seconds is too long—you can reduce costs by applying a scaling-specific cooldown period of 180 seconds to the scale-in policy.
* If your application is scaling up and down multiple times each hour, modify the Auto Scaling Groups cool-down timers and the CloudWatch Alarm Period that triggers the scale in.

**Auto scalling by policies hands on**

**Post ASG creation….select the ASG …..Automatica scaling….Add scaling policy**

Policy type: Target tracking scaling / step /simple scalling

Metric Type: AVG CPU / avg network in / avg network out / ALB request per target

Target valaue: 50 if selected CPU

Instances need: 300 [cool down period]