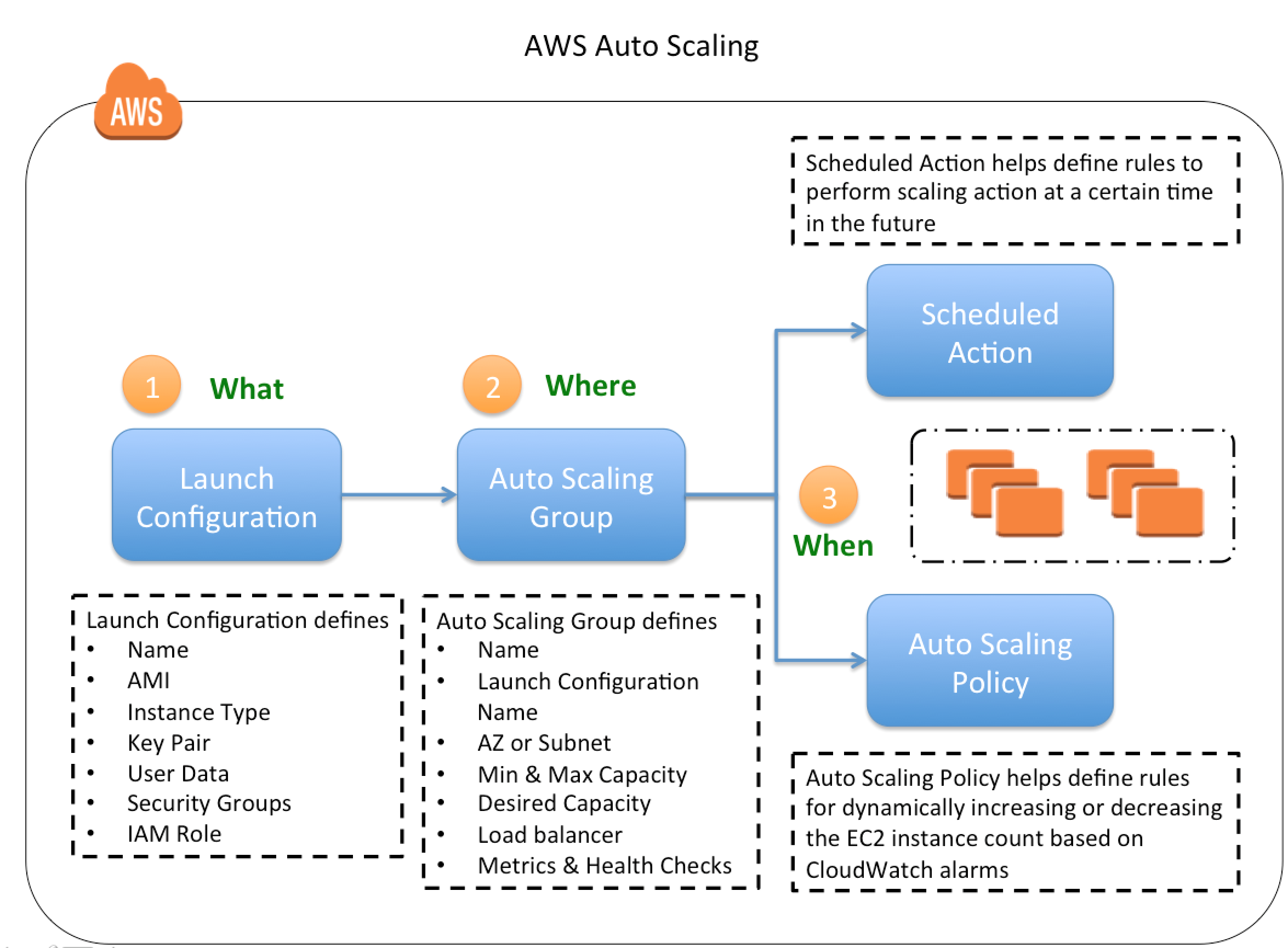
AWS Auto Scaling

# Auto Scaling Overview

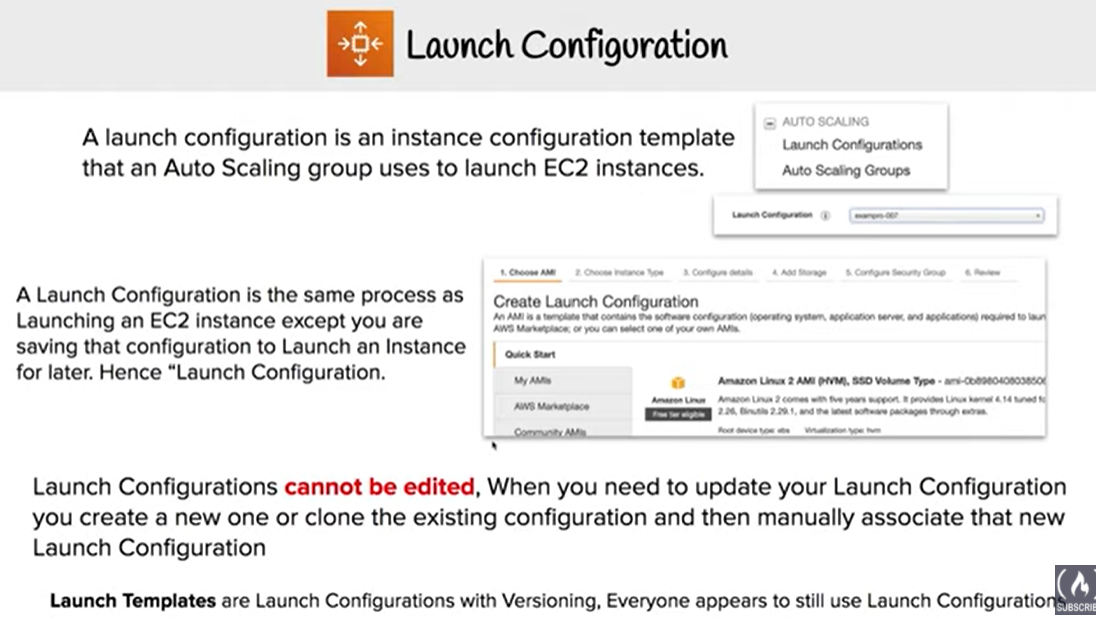
* Auto Scaling provides the ability to ensure a correct number of EC2 instances are always running to handle the load of the application
* **Auto Scaling helps to achieve better fault tolerance, better availability, and cost management**
* Auto Scaling also helps specify Scaling policies which can then be used to launch and terminate EC2 instances to handle any increase or decrease in demand on the application.
* Auto Scaling attempts to distribute instances evenly between the AZs that are enabled for the Auto Scaling group.
* **Auto Scaling does this by attempting to launch new instances in the AZ with the fewest instances. If the attempt fails, Auto Scaling attempts to launch the instances in another Availability Zone until it succeeds**

## Auto Scaling Configuration



### Launch Configuration

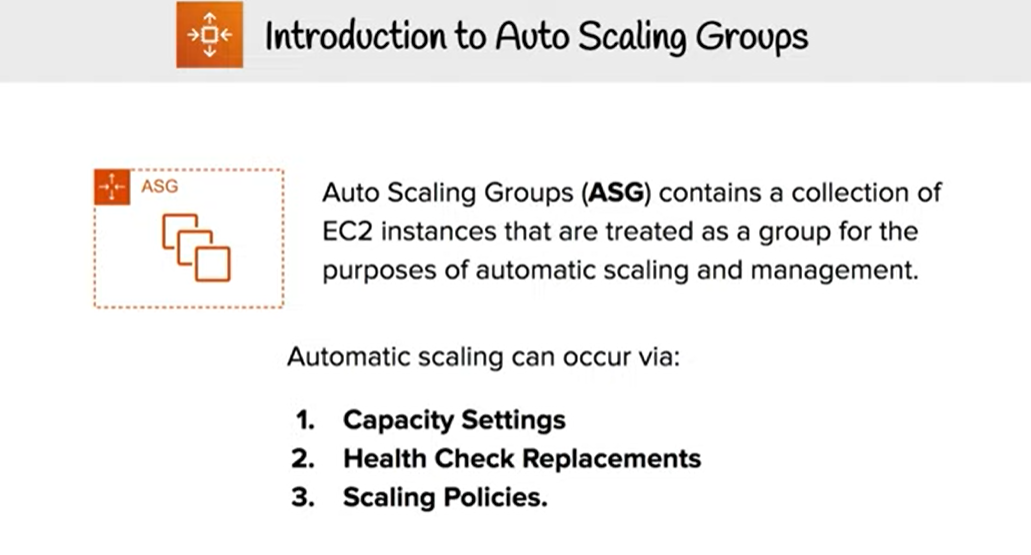
* Launch configuration is an instance configuration template that an Auto Scaling group uses to launch EC2 instances.
* Launch configuration is similar to EC2 configuration and involves selection of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping.
* Launch configuration can be associated multiple Auto Scaling groups
* **Launch configuration can’t be modified after creation and needs to be created new if any modification required**
* **Basic or detailed monitoring for the instances in the Auto Scaling group can be enabled when a launch configuration is created**.
* **By default, basic monitoring is enabled when you create the launch configuration using the AWS Management Console and detailed monitoring is enabled when you create the launch configuration using the AWS CLI or an API**



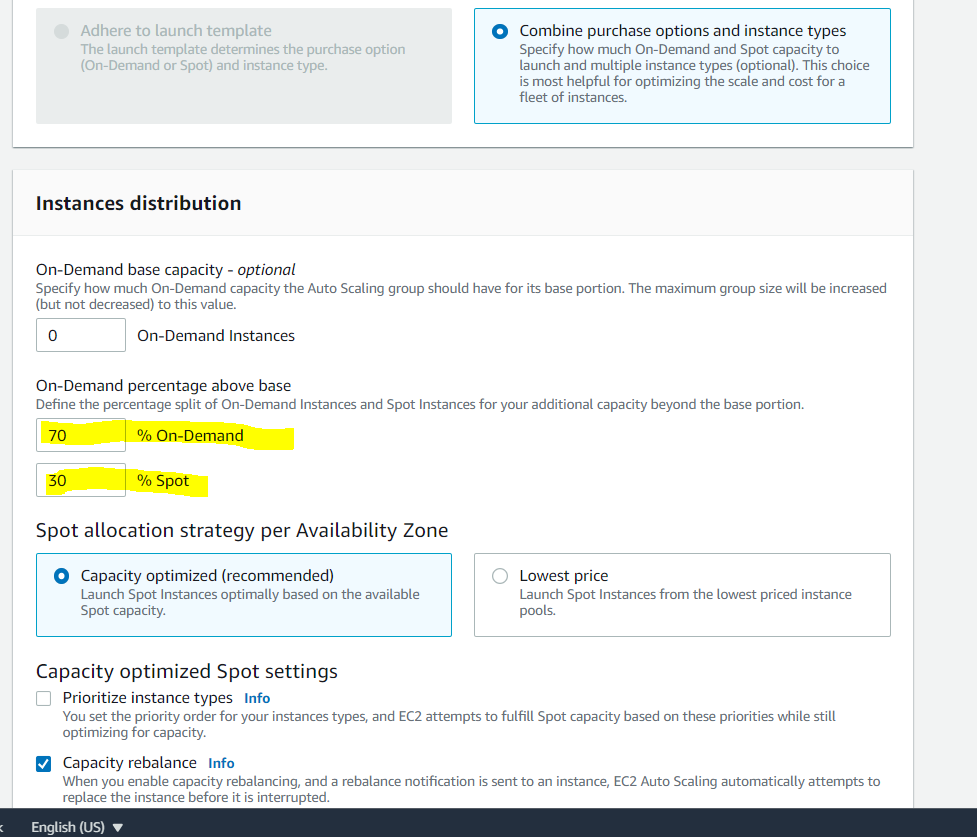
### Launch Template

* A Launch Template is similar to a launch configuration, with additional features and recommended by AWS.
* **Launch Template allows multiple versions of a template to be defined.**
* With versioning, a subset of the full set of parameters can be created and then reused to create other templates or template versions for e.g, default template that defines common configuration parameters can be created and allow the other parameters to be specified as part of another version of the same template.
* Launch Template allows selection of both **Spot and On-Demand Instances or multiple instance types.**
* Launch templates supports **EC2 Dedicated Hosts**. Dedicated Hosts are physical servers with EC2 instance capacity that are dedicated to your use.

### Auto Scaling Group – ASG

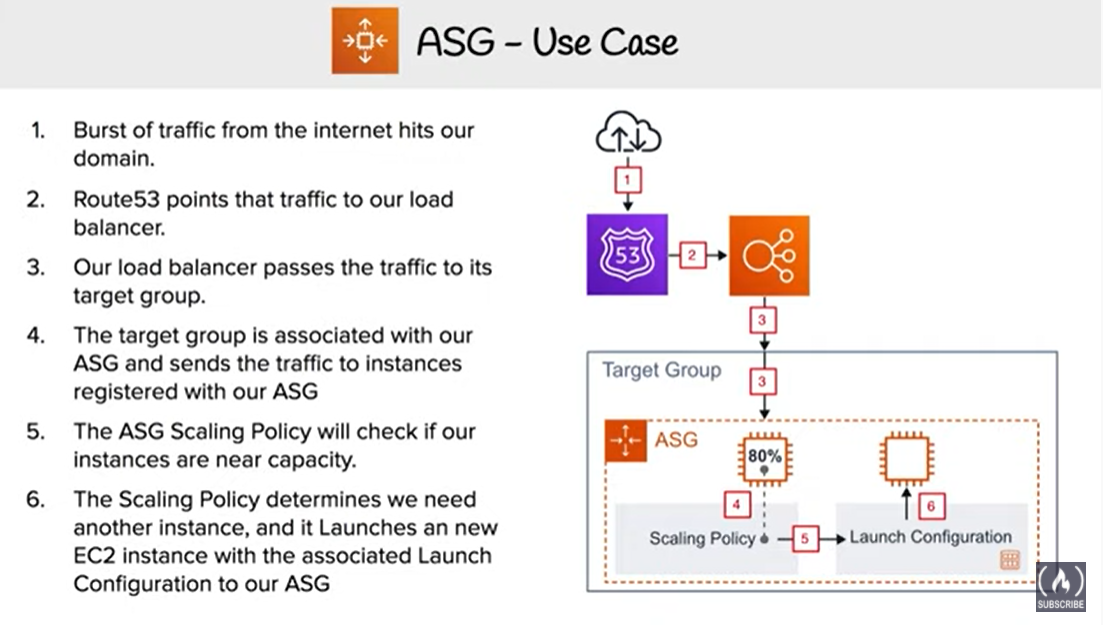


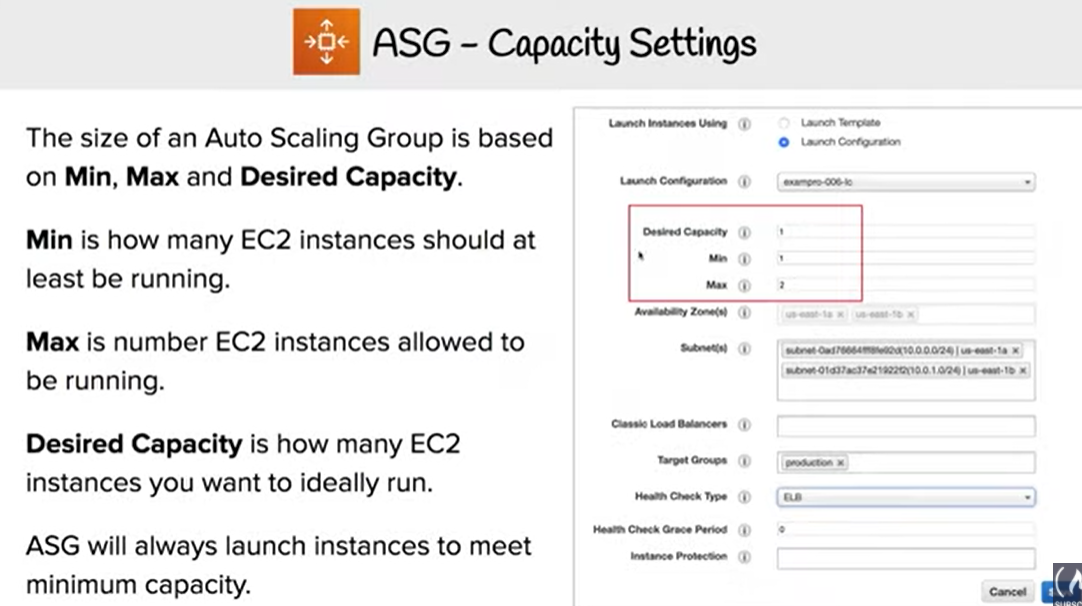
* Auto Scaling groups is the core of Auto Scaling and contains a collection of EC2 instances that share similar characteristics and are treated as a logical grouping for the purposes of instance scaling and management.
* Auto Scaling group requires
  + **Launch configuration** OR **Launch Template**
    - determine the EC2 template to use for launching the instance
  + **Minimum & Maximum capacity**
    - determine the number of instances when an autoscaling policy is applied. Number of instances cannot grow beyond this boundaries
  + **Desired capacity**
    - to determine the number of instances the ASG must maintain at all times. If missing, it equals to the minimum size.
    - Desired capacity is different from minimum capacity.
    - An Auto Scaling group’s desired capacity is the default number of instances that should be running. A group’s minimum capacity is the fewest number of instances the group can have running
  + **Availability Zones or Subnets** in which the instances will be launched.
  + **Metrics & Health Checks**
    - metrics to determine when it should launch or terminate instances and health checks to determine if the instance is healthy or not
* Auto Scaling group starts by launching a desired capacity of instances and maintains this number by performing periodic health checks.
* If an instance becomes unhealthy, Auto Scaling group terminates and launches a new instance
* Auto Scaling group can also use scaling policies to increase or decrease the number of instances automatically to meet changing demands
* An Auto Scaling group can contain EC2 instances in one or more AZs within the same region.
* **Auto Scaling group can launch On-Demand Instances, Spot Instances, or both, when configured to use a launch template**

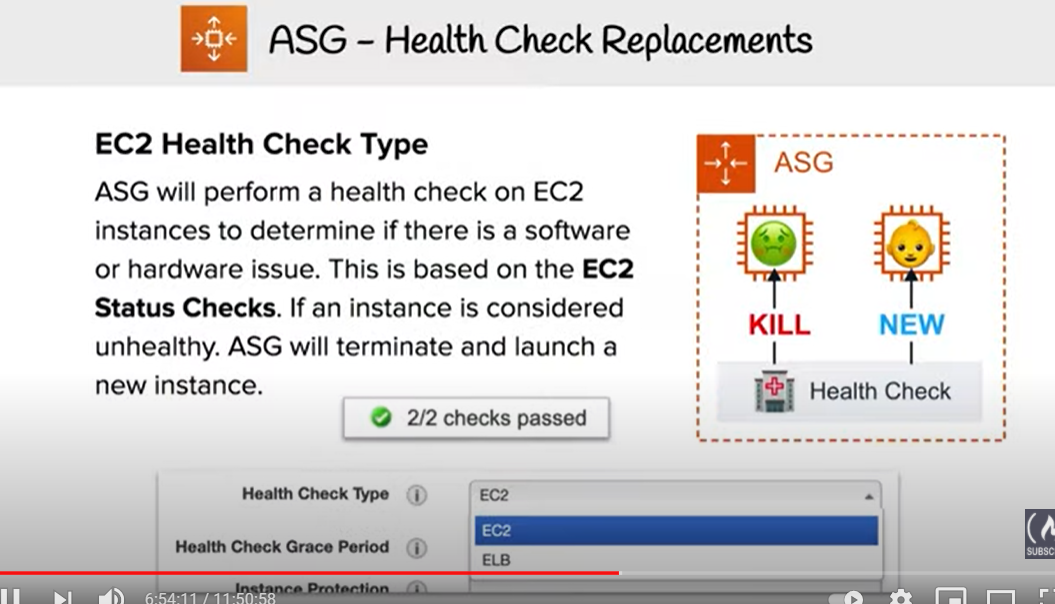


* Auto Scaling groups **cannot span multiple regions.**
* To merge separate single-zone Auto Scaling groups into a single Auto Scaling group spanning multiple AZs, rezone one of the single-zone groups into a multi-zone group, and then delete the other groups. This process works for groups with or without a load balancer, as long as the new multi-zone group is in one of the same AZs as the original single-zone groups.
* **Auto Scaling group can be associated with a single launch configuration.**
* As the Launch Configuration can’t be modified once created, only way to update the Launch Configuration for an Auto Scaling group is to create a new one and associate it with the Auto Scaling group.
* When the launch configuration for the Auto Scaling group is changed, any new instances launched use the new configuration parameters, but the existing instances are not affected.
* Auto Scaling group can be deleted from CLI, if it has no running instances else need to set the minimum and desired capacity to 0. This is handled automatically when deleting an ASG from AWS management console.

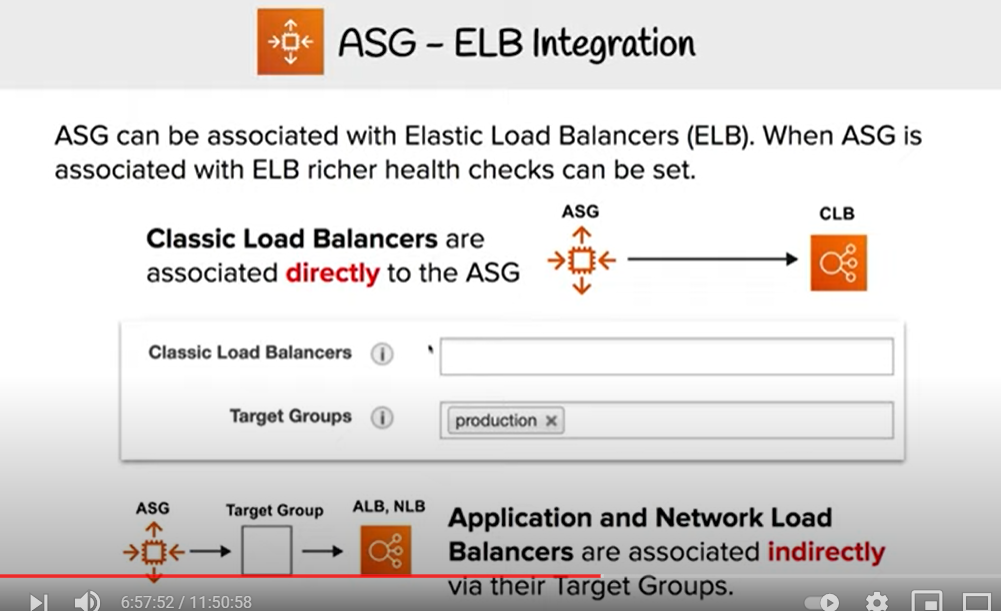
## Auto Scaling Options







### Maintain steady count of Instances

* Auto Scaling ensures a steady minimum (or desired if specified) count of Instances will always be running.
* **If an instance is found unhealthy, Auto Scaling will terminate the Instance and launch a new one**
* Auto Scaling group determines the health state of each instance by periodically checking the results of EC2 instance status checks
* **Auto Scaling group can be associated with a load balancer enabled to use the Elastic Load Balancing health check, Auto Scaling determines the health status of the instances by checking the results of both EC2 instance status and Elastic Load Balancing instance health.**
* Auto Scaling marks an instance unhealthy and launches a replacement if
  + the instance is in a state other than *running*,
  + the system status is *impaired*, or
  + Elastic Load Balancing reports the instance state as *OutOfService*.
* After an instance has been marked unhealthy as a result of an EC2 or ELB health check, it is almost immediately scheduled for replacement. It never automatically recovers its health.
* For an unhealthy instance, the instance’s health check can be changed back to healthy manually but you will encounter an error if the instance is already terminating.
* Because the interval between marking an instance unhealthy and its actual termination is so small, attempting to set an instance’s health status back to healthy is probably useful only for a suspended group
* When the instance is terminated, any associated Elastic IP addresses are disassociated and are not automatically associated with the new instance.
* Elastic IP addresses must be associated with the new instance manually.
* Similarly, when the instance is terminated, its attached EBS volumes are detached and must be attached to the new instance manually
* 

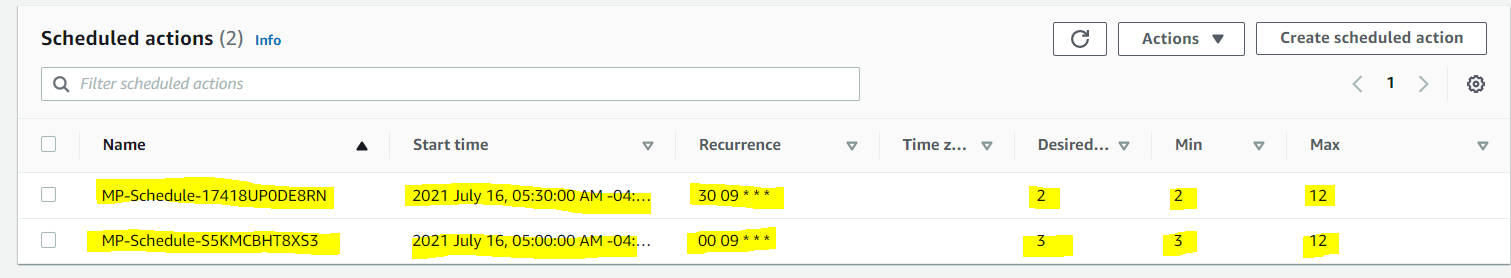
### 

### Manual scaling

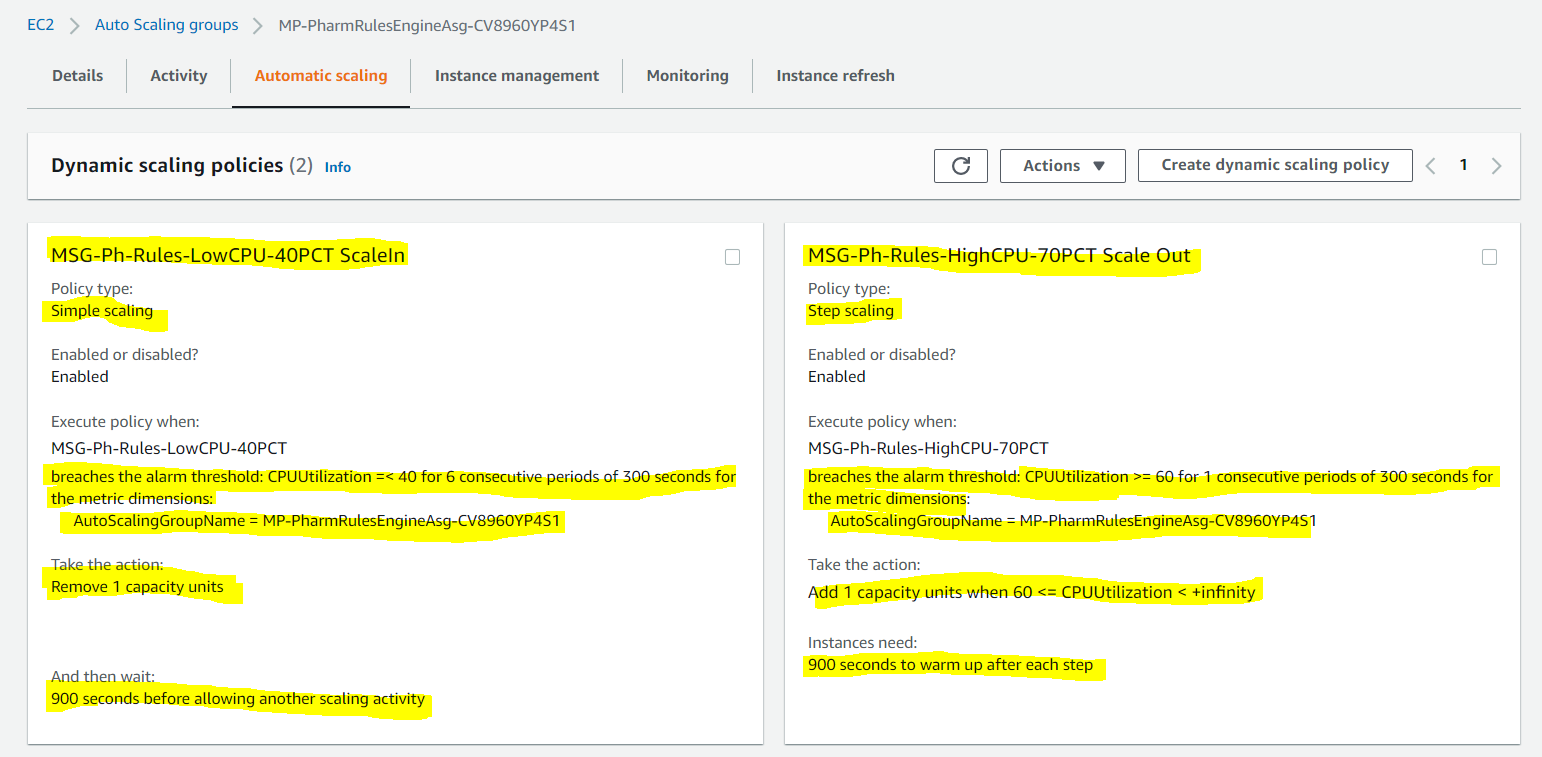
* Manual scaling can be performed by
  + Changing the desired capacity limit of the Auto Scaling group
  + Attaching/Detaching instances to the Auto Scaling group
* Attaching/Detaching of an EC2 instance can be done only if
  + Instance is in the running state.
  + AMI used to launch the instance must still exist.
  + Instance is not a member of another Auto Scaling group.
  + Instance is in the same Availability Zone as the Auto Scaling group.
  + If the Auto Scaling group is associated with a load balancer, the instance and the load balancer must both be in the same VPC.
* Auto Scaling increases the desired capacity of the group by the number of instances being attached. But if the number of instances being attached plus the desired capacity exceeds the maximum size, the request fails.
* When Detaching instances, an option to decrement the desired capacity for the Auto Scaling group by the number of instances being detached is provided. If chosen not to decrement the capacity, Auto Scaling launches new instances to replace the ones that you detached.
* **If an instance is detached from an Auto Scaling group that is also registered with a load balancer, the instance is deregistered from the load balancer. If connection draining is enabled for the load balancer, Auto Scaling waits for the in-flight requests to complete.**

### Scheduled scaling

* Scaling based on a schedule allows you to scale your application in response to **predictable load changes** for e.g. last day of the month, last day of an financial year
* Scheduled scaling requires configuration of Scheduled actions, which tells Auto Scaling to perform a scaling action at certain time in future, with the start time at which the scaling action should take effect, and the new minimum, maximum, and desired size the group should have
* Auto Scaling guarantees the order of execution for scheduled actions within the same group, but not for scheduled actions across groups
* Multiple Scheduled Actions can be specified but should have **unique** time value and they **cannot** have overlapping time scheduled which will lead to its rejection
* **Cooldown periods are not supported.**



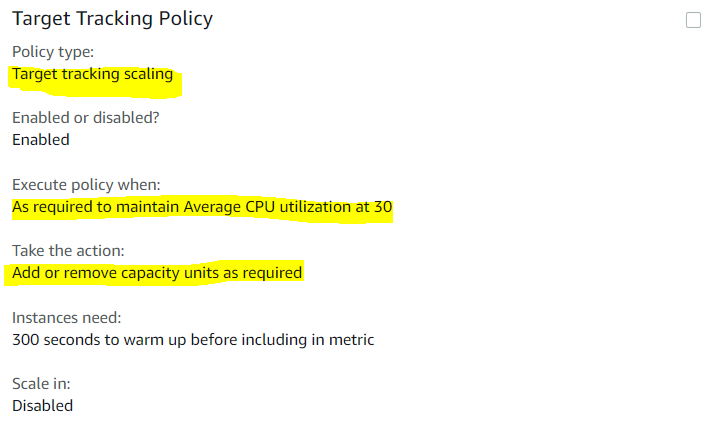
### Dynamic scaling



* Allows you to scale automatically in response to the changing demand for e.g. scale out in case CPU utilization of the instance goes above 70% and scale in when the CPU utilization goes below 30%
* Auto Scaling group uses a combination of **alarms & policies** to determine when the conditions for scaling are met.
  + An alarm is an object that watches over a single metric over a specified time period. When the value of the metric breaches the defined threshold, for the number of specified time periods the alarm performs one or more actions (such as sending messages to Auto Scaling).
  + A policy is a set of instructions that tells Auto Scaling how to respond to alarm messages.
* Dynamic scaling process works as below
  + CloudWatch monitors the specified metrics for all the instances in the Auto Scaling group
  + Changes are reflected in the metrics as the demand grows or shrinks
  + When the change in the metrics breaches the threshold of the CloudWatch alarm, the CloudWatch alarm performs an action. Depending on the breach, the action is a message sent to either the scale-in policy or the scale-out policy
  + After the Auto Scaling policy receives the message, Auto Scaling performs the scaling activity for the Auto Scaling group.
  + This process continues until you delete either the scaling policies or the Auto Scaling group.
* When a scaling policy is executed, if the capacity calculation produces a number outside of the minimum and maximum size range of the group, EC2 Auto Scaling ensures that the new capacity never goes outside of the minimum and maximum size limits.
* When the desired capacity reaches the maximum size limit, scaling out stops. If demand drops and capacity decreases, EC2 Auto Scaling can scale out again.

#### **Scaling Policy Types**

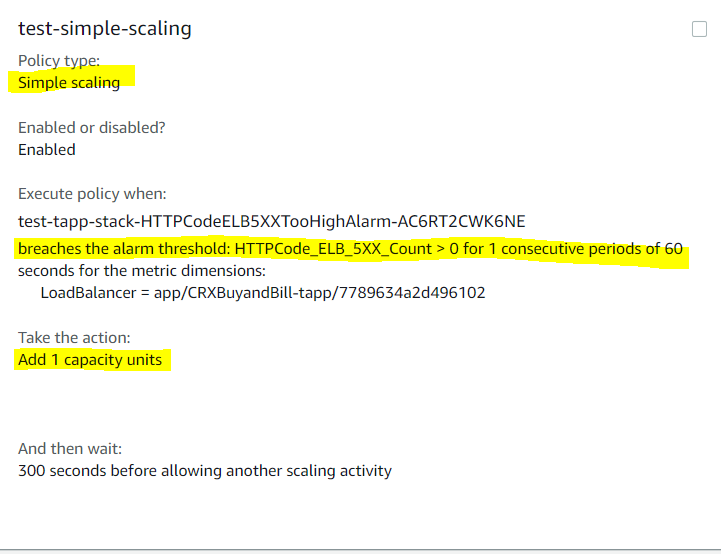
* **Target tracking scaling**
  + Increase or decrease the current capacity of the group based on a target value for a specific metric.



Diagram

Description automatically generated

* **Simple scaling**
  + Increase or decrease the current capacity of the group based on a single scaling adjustment.



Diagram, timeline

Description automatically generated

* **Step scaling**
  + Increase or decrease the current capacity of the group based on a set of scaling adjustments, known as step adjustments, that vary based on the size of the alarm breach.

Graphical user interface, text, application, email

Description automatically generated

Timeline

Description automatically generated

### 

### Multiple Policies

* ASG can have more than one scaling policy attached to it any given time.
* Each Auto Scaling group would have at least two policies: one to scale the architecture out and another to scale the architecture in.
* If an ASG has multiple policies, there is always a chance that both policies can instruct the Auto Scaling to Scale Out or Scale In at the same time.
* When this situations occur, Auto Scaling chooses the policy that has the **greatest impact** i.e. **provides the largest capacity for both scale out and scale in**on the Auto Scaling group for e.g. if two policies are triggered at the same time and Policy 1 instructs to scale out the instance by 1 while Policy 2 instructs to scale out the instances by 2, Auto Scaling will use the Policy 2 and scale out the instances by 2 as it has a greater impact

### Auto Scaling Cooldown

* Auto Scaling Cooldown period is a configurable setting for the ASG that helps to ensure that Auto Scaling doesn’t launch or terminate additional instances before the previous scaling activity takes effect and allows the newly launched instances to start handling traffic and reduce load
* When ASG dynamically scales using a simple scaling policy and launches an instance, Auto Scaling suspends the scaling activities for the cooldown period (default 300 seconds) to complete before resuming scaling activities
* Example Use Case
  + You configure an scale out alarm to increase the capacity if the CPU utilization increases more then 80%
  + An CPU spikes occurs and cause the alarm to be triggered, Auto Scaling launches a new instance
  + However, it would take time for the newly launched instance to be configured, instantiated and started, lets say 5 mins
  + Without a cooldown period, if an other CPU spikes occurs Auto Scaling would launch a new instance again and this would continue for 5 mins till the previously launched instance is up and running and started handling traffic
  + With a cooldown period, Auto Scaling would suspend the activity for the specified time period enabling the newly launched instance to start handling traffic and reduce load
  + After the cooldown period, Auto scaling resumes to act on the alarms
* **When manually scaling the ASG, the default is not to wait for the cooldown period, but can be overrided to honor the cooldown period**
* Note that if an instance becomes unhealthy, Auto Scaling does not wait for the cooldown period to complete before replacing the unhealthy instance.
* Cooldown periods are automatically applied to dynamic scaling activities for simple scaling policies and is **not** supported for step scaling policies.

## Termination Policy

* Termination policy helps the Auto Scaling to decide which instances it should terminate first when Auto Scaling automatically scales in.
* Auto Scaling specifies a default termination policy and also provides the ability to create a customized one.

### Default Termination Policy

Default termination policy helps ensure that the network architecture spans AZs evenly and instances are selected for termination as follows :-

1. **Selection of Availability Zone**
   * selects the AZ, in multiple AZs environment, with the **most instances** and at least one instance that is not protected from scale in.
   * selects the AZ with instances that use the **oldest launch configuration**, if there are more than one AZ with same number of instances
2. **Selection of an Instance in the Availability Zone**
   * terminates the **unprotected instance using the oldest launch configuration**, if one exists.
   * terminates unprotected instances **closest to the next billing hour,** If multiple instances with oldest launch configuration. This helps in maximizing the use of the EC2 instances that have an hourly charge while minimizing the number of hours billed for EC2 usage.
   * terminates instances at **random**, if more than one unprotected instance closest to the next billing hour

### Customized Termination Policy

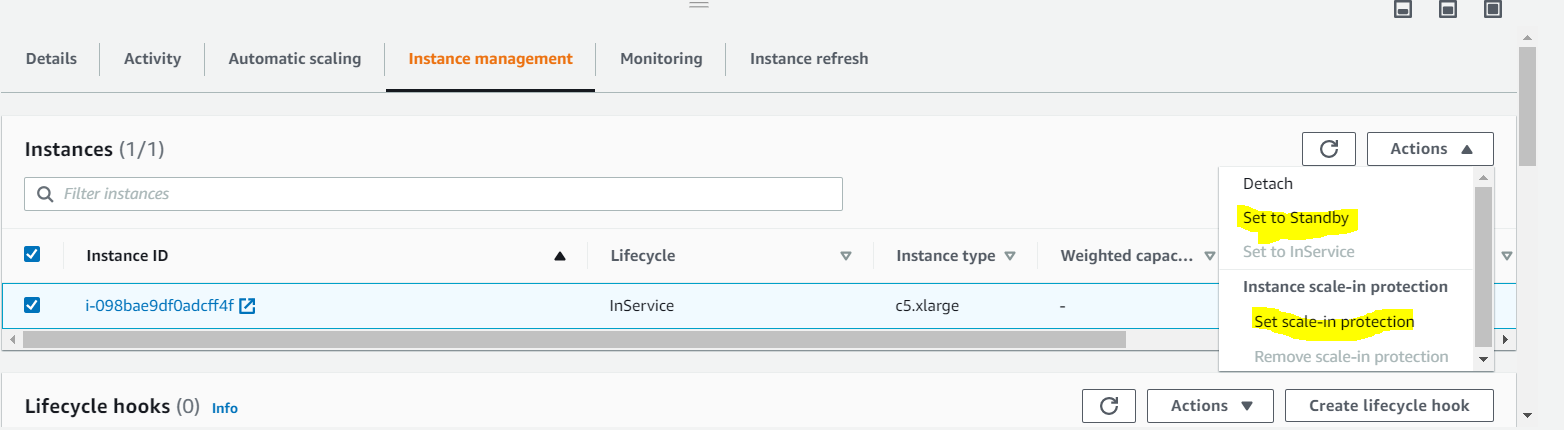
1. Auto Scaling first assesses the AZs for any imbalance. If an AZ has more instances than the other AZs that are used by the group, then it applies the specified termination policy on the instances from the imbalanced AZ
2. If the Availability Zones used by the group are balanced, then Auto Scaling applies the termination policy that you specified.
3. Following Customized Termination policies are supported
   1. **OldestInstance –**terminates the oldest instance in the group and can be useful to upgrade to new instance types
   2. **NewestInstance –**terminates the newest instance in the group and can be useful when testing a new launch configuration
   3. **OldestLaunchConfiguration –**terminates instances that have the oldest launch configuration
   4. **OldestLaunchTemplate –**terminates instances that have the oldest launch template
   5. **ClosestToNextInstanceHour –** terminates instances that are closest to the next billing hour and helps to maximize the use of your instances and manage costs.
   6. **Default –**terminates as per the default termination policy

### Instance Protection

* Instance protection controls whether Auto Scaling can terminate a particular instance or not.
* Instance protection can be enabled on a Auto Scaling group or an individual instance as well, at any time
* Instances launched within an Auto Scaling group with Instance protection enabled would inherit the property.
* Instance protection starts as soon as the instance is InService and if the Instance is detached, it loses its Instance protection
* If all instances in an ASG are protected from termination during scale in and a scale-in event occurs, it can’t terminate any instance and will decrement the desired capacity.
* Instance protection does not protect for the below cases
  + **Manual termination** through the EC2 console, the terminate-instances command, or the TerminateInstances API.
  + Termination if it **fails health checks** and must be replaced
  + **Spot instances** in an Auto Scaling group from interruption

## Standby State

Auto Scaling allows putting the InService instance in the Standby state during which the instance is still a part of the ASG but does not serve any requests. This can be used to either troubleshoot an instance or update an instance and return the instance back to service



* An instance can be put into Standby state and it will continue to remain in the Standby state unless exited
* Auto Scaling, by default, decrements the desired capacity for the group and prevents it from launching a new instance. If no decrement is selected, it would launch a new instance
* When the instance is in the standby state, instance can be updated or used for troubleshooting
* If a load balancer is associated with Auto Scaling, the instance is automatically deregistered when the instance is in Standby state and registered again when the instance exits the Standby state

## Suspension

* Auto Scaling processes can be suspended and then resumed. This can be very useful to investigate a configuration problem or debug an issue with the application, without triggering the Auto Scaling process.
* Auto Scaling also performs **Administrative Suspension**where it would suspend processes for ASGs, if the Auto Scaling groups that have been trying to launch instances for over 24 hours but have not succeeded in launching any instances.
* Auto Scaling processes include
  + **Launch** – Adds a new EC2 instance to the group, increasing its capacity.
  + **Terminate** – Removes an EC2 instance from the group, decreasing its capacity.
  + **HealthCheck** -Checks the health of the instances.
  + **ReplaceUnhealthy** – Terminates instances that are marked as unhealthy and subsequently creates new instances to replace them.
  + **AlarmNotification** – Accepts notifications from CloudWatch alarms that are associated with the group. If suspended,  Auto Scaling does not automatically execute policies that would be triggered by an alarm
  + **ScheduledActions** – Performs scheduled actions that you create.
  + **AddToLoadBalancer** – Adds instances to the load balancer when they are launched.
  + **AZRebalance** – Balances the number of EC2 instances in the group across the Availability Zones in the region.
    - If an AZ either is removed from the ASG or becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected AZ before terminating the unhealthy or unavailable instances
    - When the unhealthy AZ returns to a healthy state, Auto Scaling automatically redistributes the instances evenly across the Availability Zones for the group.
    - Note that if you suspend AZRebalance and a scale out or scale in event occurs, Auto Scaling still tries to balance the Availability Zones for e.g. during scale out, it launches the instance in the Availability Zone with the fewest instances.
    - If you suspend Launch, AZRebalance neither launches new instances nor terminates existing instances. This is because AZRebalance terminates instances only after launching the replacement instances.
    - If you suspend Terminate, the ASG can grow up to 10% larger than its maximum size, because Auto Scaling allows this temporarily during rebalancing activities. If it cannot terminate instances, your ASG could remain above its maximum size until the Terminate process is resumed

# Auto Scaling Lifecycle

* Instances launched through Auto Scaling group have a a different lifecycle then that of other EC2 instances
* Auto Scaling lifecycle starts when the Auto Scaling group launches an instance and puts it into service.
* Auto Scaling lifecycle ends when the instance is terminated either by the user , or the Auto Scaling group takes it out of service and terminates it
* AWS charges for the instances as soon as they are launched, including the time it is not in InService

## Auto Scaling Lifecycle Transition

# Auto Scaling Group Lifecycle

### Auto Scaling Lifecycle Hooks

* Auto Scaling Lifecycle hooks enable performing custom actions by pausing instances as an Auto Scaling group launches or terminates them
* Each Auto Scaling group can have multiple lifecycle hooks. However, there is a limit on the number of hooks per Auto Scaling group
* Auto Scaling scale out event flow
  + Instances start in the **Pending** state
  + If an autoscaling:EC2\_INSTANCE\_LAUNCHINGlifecycle hookis added, the state is moved to **Pending:Wait**
  + After the lifecycle action is completed, instances enter to **Pending:Proceed**
  + When the instances are fully configured, they are attached to the Auto Scaling group and moved to the **InService** state
* Auto Scaling scale in event flow
  + Instances are detached from the Auto Scaling group and enter the **Terminating** state.
  + If an autoscaling:EC2\_INSTANCE\_TERMINATING lifecycle hookis added, the state is moved to **Terminating:Wait**
  + After the lifecycle action is completed, the instances enter the **Terminating:Proceed** state.
  + When the instances are fully terminated, they enter the **Terminated** state.
* During the scale out and scale in events, instances are put into a wait state  (Pending:Wait or Terminating:Wait) and is paused until either a continue action happens or the timeout period ends.
* By default, the instance remains in a wait state for one hour, which can be extended by restarting the timeout period by recoding a heartbeat.
* If the task finishes before the timeout period ends, the lifecycle action can be marked completed and it continues the launch or termination process.
* After the wait period the Auto Scaling group continues the launch or terminate process (Pending:Proceed or Terminating:Proceed)
  + CloudWatch Events target to invoke a Lambda function when a lifecycle action occurs. Event contains information about the instance that is launching or terminating, and a token that can be used to control the lifecycle action.
  + Notification target (CloudWatch events, SNS, SQS) for the lifecycle hook which receives the message from EC2 Auto Scaling.The message contains information about the instance that is launching or terminating, and a token that can be used to control the lifecycle action
  + Create a script that runs on the instance as the instance starts. The script can control the lifecycle action using the ID of the instance on which it runs.Custom action can be implemented using

## Auto Scaling Lifecycle Hooks Considerations

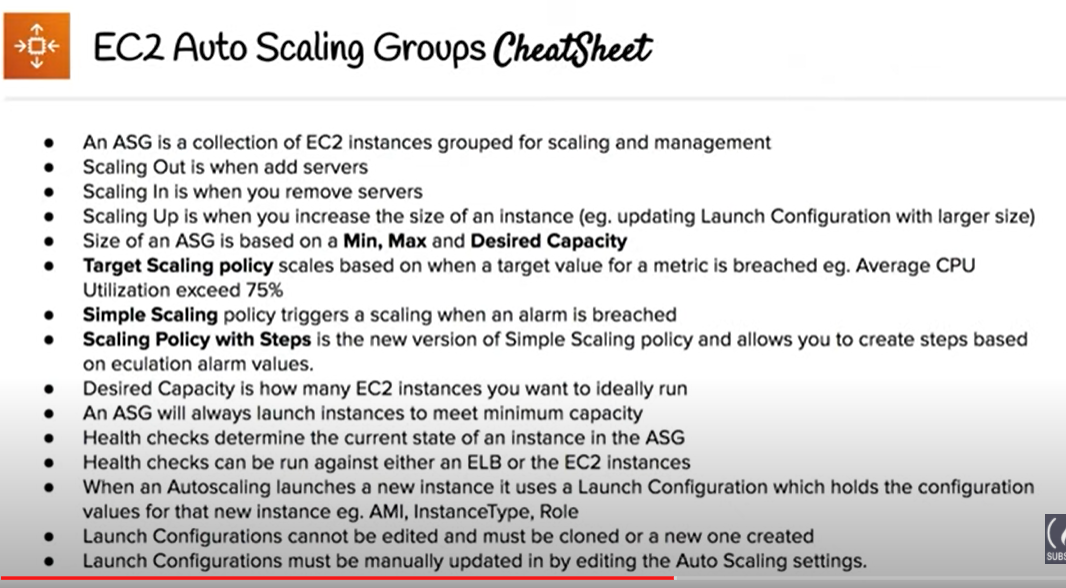
* **Keeping Instances in a Wait State**
  + Instances remain in a wait state for a finite period of time.
  + Default being 1 hour (3600 seconds) with max being 48 hours or 100 times the heartbeat timeout, whichever is smaller.
  + Time can be adjusted using
    - complete-lifecycle-action (CompleteLifecycleAction) command to continue to the next state if finishes before the timeout period ends
    - put-lifecycle-hook command, the –heartbeat-timeout parameter to set the heartbeat timeout for the lifecycle hook during its creation
    - Restart the timeout period by recording a heartbeat, using the record-lifecycle-action-heartbeat (RecordLifecycleActionHeartbeat) command
* **Cooldowns and Custom Actions**
  + Cooldown period helps ensure that the Auto Scaling group does not launch or terminate more instances than needed
  + Cooldown period starts when the instance enters the InService state. Any suspended scaling actions resume after cooldown period expires
* **Health Check Grace Period**
  + Health check grace period does not start until the lifecycle hook completes and the instance enters the InService state
* **Lifecycle Action Result**
  + Result of lifecycle hook is either ABANDON or CONTINUE
  + If the instance is launching,
    - CONTINUE indicates a successful action, and the instance can be put into service.
    - ABANDON indicates the custom actions were unsuccessful, and that the instance can be terminated.
  + If the instance is terminating,
    - ABANDON and CONTINUE allow the instance to terminate.
    - However, ABANDON stops any remaining actions from other lifecycle hooks, while CONTINUE allows them to complete
* **Spot Instances**
  + Lifecycle hooks can be used with Spot Instances. However, a lifecycle hook does not prevent an instance from terminating due to a change in the Spot Price, which can happen at any time

### Enter and Exit Standby

* Instance in an InService state can be moved toStandby state.
* Standby state enables you to remove the instance from service, troubleshoot or make changes to it, and then put it back into service.
* Instances in a Standby state continue to be managed by the Auto Scaling group. However, they are not an active part of the application until they are put back into service.

Text

Description automatically generated



## AWS Certification Exam Practice Questions

1. A user is trying to setup a scheduled scaling activity using Auto Scaling. The user wants to setup the recurring schedule. Which of the below mentioned parameters is not required in this case?
   1. **Maximum size**
   2. Auto Scaling group name
   3. End time
   4. Recurrence value
2. A user has configured Auto Scaling with 3 instances. The user had created a new AMI after updating one of the instances. If the user wants to terminate two specific instances to ensure that Auto Scaling launches an instances with the new launch configuration, which command should he run?
   1. as-delete-instance-in-auto-scaling-group <Instance ID> –no-decrement-desired-capacity
   2. as-terminate-instance-in-auto-scaling-group <Instance ID> –update-desired-capacity
   3. as-terminate-instance-in-auto-scaling-group <Instance ID> –decrement-desired-capacity
   4. **as-terminate-instance-in-auto-scaling-group <Instance ID> –no-decrement-desired-capacity**
3. A user is planning to scale up an application by 8 AM and scale down by 7 PM daily using Auto Scaling. What should the user do in this case?
   1. Setup the scaling policy to scale up and down based on the CloudWatch alarms
   2. User should increase the desired capacity at 8 AM and decrease it by 7 PM manually
   3. User should setup a batch process which launches the EC2 instance at a specific time
   4. **Setup scheduled actions to scale up or down at a specific time**
4. An organization has setup Auto Scaling with ELB. Due to some manual error, one of the instances got rebooted. Thus, it failed the Auto Scaling health check. Auto Scaling has marked it for replacement. How can the system admin ensure that the instance does not get terminated?
   1. Update the Auto Scaling group to ignore the instance reboot event
   2. It is not possible to change the status once it is marked for replacement
   3. Manually add that instance to the Auto Scaling group after reboot to avoid replacement
   4. **Change the health of the instance to healthy using the Auto Scaling commands**
5. A user has configured Auto Scaling with the minimum capacity as 2 and the desired capacity as 2. The user is trying to terminate one of the existing instance with the command: as-terminate-instance-in-auto-scaling-group<Instance ID> –decrement-desired-capacity. What will Auto Scaling do in this scenario?
   1. Terminates the instance and does not launch a new instance
   2. Terminates the instance and updates the desired capacity to 1
   3. Terminates the instance and updates the desired capacity & minimum size to 1
   4. **Throws an error**
6. An organization has configured Auto Scaling for hosting their application. The system admin wants to understand the Auto Scaling health check process. If the instance is unhealthy, Auto Scaling launches an instance and terminates the unhealthy instance. What is the order execution?
   1. Auto Scaling launches a new instance first and then terminates the unhealthy instance
   2. Auto Scaling performs the launch and terminate processes in a random order
   3. Auto Scaling launches and terminates the instances simultaneously
   4. **Auto Scaling terminates the instance first and then launches a new instance**
7. A user has configured ELB with Auto Scaling. The user suspended the Auto Scaling terminate process only for a while. What will happen to the availability zone rebalancing process (AZRebalance) during this period?
   1. Auto Scaling will not launch or terminate any instances
   2. **Auto Scaling will allow the instances to grow more than the maximum size**
   3. Auto Scaling will keep launching instances till the maximum instance size
   4. It is not possible to suspend the terminate process while keeping the launch active
8. An organization has configured Auto Scaling with ELB. There is a memory issue in the application which is causing CPU utilization to go above 90%. The higher CPU usage triggers an event for Auto Scaling as per the scaling policy. If the user wants to find the root cause inside the application without triggering a scaling activity, how can he achieve this?
   1. Stop the scaling process until research is completed
   2. It is not possible to find the root cause from that instance without triggering scaling
   3. Delete Auto Scaling until research is completed
   4. **Suspend the scaling process until research is completed**
9. A user has configured ELB with Auto Scaling. The user suspended the Auto Scaling Alarm Notification (which notifies Auto Scaling for CloudWatch alarms) process for a while. What will Auto Scaling do during this period?
   1. AWS will not receive the alarms from CloudWatch
   2. **AWS will receive the alarms but will not execute the Auto Scaling policy**
   3. Auto Scaling will execute the policy but it will not launch the instances until the process is resumed
   4. It is not possible to suspend the AlarmNotification process
10. An organization has configured two single availability zones. The Auto Scaling groups are configured in separate zones. The user wants to merge the groups such that one group spans across multiple zones. How can the user configure this?
    1. Run the command as-join-auto-scaling-group to join the two groups
    2. **Run the command as-update-auto-scaling-group to configure one group to span across zones and delete the other group**
    3. Run the command as-copy-auto-scaling-group to join the two groups
    4. Run the command as-merge-auto-scaling-group to merge the groups
11. An organization has configured Auto Scaling with ELB. One of the instance health check returns the status as Impaired to Auto Scaling. What will Auto Scaling do in this scenario?
    1. Perform a health check until cool down before declaring that the instance has failed
    2. **Terminate the instance and launch a new instance**
    3. Notify the user using SNS for the failed state
    4. Notify ELB to stop sending traffic to the impaired instance
12. A user has setup an Auto Scaling group. The group has failed to launch a single instance for more than 24 hours. What will happen to Auto Scaling in this condition
    1. Auto Scaling will keep trying to launch the instance for 72 hours
    2. **Auto Scaling will suspend the scaling process**
    3. Auto Scaling will start an instance in a separate region
    4. The Auto Scaling group will be terminated automatically
13. A user is planning to setup infrastructure on AWS for the Christmas sales. The user is planning to use Auto Scaling based on the schedule for proactive scaling. What advise would you give to the user?
    1. It is good to schedule now because if the user forgets later on it will not scale up
    2. The scaling should be setup only one week before Christmas
    3. **Wait till end of November before scheduling the activity**
    4. It is not advisable to use scheduled based scaling
14. A user is trying to setup a recurring Auto Scaling process. The user has setup one process to scale up every day at 8 am and scale down at 7 PM. The user is trying to setup another recurring process which scales up on the 1st of every month at 8 AM and scales down the same day at 7 PM. What will Auto Scaling do in this scenario
    1. Auto Scaling will execute both processes but will add just one instance on the 1st
    2. Auto Scaling will add two instances on the 1st of the month
    3. Auto Scaling will schedule both the processes but execute only one process randomly
    4. **Auto Scaling will throw an error since there is a conflict in the schedule of two separate Auto Scaling Processes**
15. A sys admin is trying to understand the Auto Scaling activities. Which of the below mentioned processes is not performed by Auto Scaling?
    1. **Reboot Instance**
    2. Schedule Actions
    3. Replace Unhealthy
    4. Availability Zone Re-Balancing
16. You have started a new job and are reviewing your company’s infrastructure on AWS. You notice one web application where they have an Elastic Load Balancer in front of web instances in an Auto Scaling Group. When you check the metrics for the ELB in CloudWatch you see four healthy instances in Availability Zone (AZ) A and zero in AZ B. There are zero unhealthy instances. What do you need to fix to balance the instances across AZs?
    1. Set the ELB to only be attached to another AZ
    2. **Make sure Auto Scaling is configured to launch in both AZs**
    3. Make sure your AMI is available in both AZs
    4. Make sure the maximum size of the Auto Scaling Group is greater than 4
17. You have been asked to leverage Amazon VPC EC2 and SQS to implement an application that submits and receives millions of messages per second to a message queue. You want to ensure your application has sufficient bandwidth between your EC2 instances and SQS. Which option will provide the most scalable solution for communicating between the application and SQS?
    1. Ensure the application instances are properly configured with an Elastic Load Balancer
    2. Ensure the application instances are launched in private subnets with the EBS-optimized option enabled
    3. Ensure the application instances are launched in public subnets with the associate-public-IP-address=trueoption enabled
    4. **Launch application instances in private subnets with an Auto Scaling group and Auto Scaling triggers configured to watch the SQS queue size**
18. You have decided to change the Instance type for instances running in your application tier that are using Auto Scaling. In which area below would you change the instance type definition?
    1. **Auto Scaling launch configuration**
    2. Auto Scaling group
    3. Auto Scaling policy
    4. Auto Scaling tags
19. A user is trying to delete an Auto Scaling group from CLI. Which of the below mentioned steps are to be performed by the user?
    1. Terminate the instances with the ec2-terminate-instance command
    2. Terminate the Auto Scaling instances with the as-terminate-instance command
    3. **Set the minimum size and desired capacity to 0**
    4. There is no need to change the capacity. Run the as-delete-group command and it will reset all values to 0
20. A user has created a web application with Auto Scaling. The user is regularly monitoring the application and he observed that the traffic is highest on Thursday and Friday between 8 AM to 6 PM. What is the best solution to handle scaling in this case?
    1. Add a new instance manually by 8 AM Thursday and terminate the same by 6 PM Friday
    2. **Schedule Auto Scaling to scale up by 8 AM Thursday and scale down after 6 PM on Friday**
    3. Schedule a policy which may scale up every day at 8 AM and scales down by 6 PM
    4. Configure a batch process to add a instance by 8 AM and remove it by Friday 6 PM
21. A user has configured the Auto Scaling group with the minimum capacity as 3 and the maximum capacity as 5. When the user configures the AS group, how many instances will Auto Scaling launch?
    1. **3**
    2. 0
    3. 5
    4. 2
22. A sys admin is maintaining an application on AWS. The application is installed on EC2 and user has configured ELB and Auto Scaling. Considering future load increase, the user is planning to launch new servers proactively so that they get registered with ELB. How can the user add these instances with Auto Scaling?
    1. **Increase the desired capacity of the Auto Scaling group**
    2. Increase the maximum limit of the Auto Scaling group
    3. Launch an instance manually and register it with ELB on the fly
    4. Decrease the minimum limit of the Auto Scaling group
23. In reviewing the auto scaling events for your application you notice that your application is scaling up and down multiple times in the same hour. What design choice could you make to optimize for the cost while preserving elasticity? Choose 2 answers.
    1. **Modify the Amazon CloudWatch alarm period that triggers your auto scaling scale down policy.**
    2. Modify the Auto scaling group termination policy to terminate the oldest instance first.
    3. Modify the Auto scaling policy to use scheduled scaling actions.
    4. **Modify the Auto scaling group cool down timers.**
    5. Modify the Auto scaling group termination policy to terminate newest instance first.
24. You have a business critical two tier web app currently deployed in two availability zones in a single region, using Elastic Load Balancing and Auto Scaling. The app depends on synchronous replication (very low latency connectivity) at the database layer. The application needs to remain fully available even if one application Availability Zone goes off-line, and Auto scaling cannot launch new instances in the remaining Availability Zones. How can the current architecture be enhanced to ensure this? **[PROFESSIONAL]**
    1. Deploy in two regions using Weighted Round Robin (WRR), with Auto Scaling minimums set for 100% peak load per region.
    2. **Deploy in three AZs, with Auto Scaling minimum set to handle 50% peak load per zone.**
    3. Deploy in three AZs, with Auto Scaling minimum set to handle 33% peak load per zone. (Loss of one AZ will handle only 66% if the autoscaling also fails)
    4. Deploy in two regions using Weighted Round Robin (WRR), with Auto Scaling minimums set for 50% peak load per region.
25. A user has created a launch configuration for Auto Scaling where CloudWatch detailed monitoring is disabled. The user wants to now enable detailed monitoring. How can the user achieve this?
    1. Update the Launch config with CLI to set InstanceMonitoringDisabled = false
    2. The user should change the Auto Scaling group from the AWS console to enable detailed monitoring
    3. Update the Launch config with CLI to set InstanceMonitoring.Enabled = true
    4. **Create a new Launch Config with detail monitoring enabled and update the Auto Scaling group**
26. A user has created an Auto Scaling group with default configurations from CLI. The user wants to setup the CloudWatch alarm on the EC2 instances, which are launched by the Auto Scaling group. The user has setup an alarm to monitor the CPU utilization every minute. Which of the below mentioned statements is true?
    1. It will fetch the data at every minute but the four data points [corresponding to 4 minutes] will not have value since the EC2 basic monitoring metrics are collected every five minutes
    2. **It will fetch the data at every minute as detailed monitoring on EC2 will be enabled by the default launch configuration of Auto Scaling**
    3. The alarm creation will fail since the user has not enabled detailed monitoring on the EC2 instances
    4. The user has to first enable detailed monitoring on the EC2 instances to support alarm monitoring at every minute
27. A customer has a website which shows all the deals available across the market. The site experiences a load of 5 large EC2 instances generally. However, a week before Thanksgiving vacation they encounter a load of almost 20 large instances. The load during that period varies over the day based on the office timings. Which of the below mentioned solutions is cost effective as well as help the website achieve better performance?
    1. Keep only 10 instances running and manually launch 10 instances every day during office hours.
    2. **Setup to run 10 instances during the pre-vacation period and only scale up during the office time by launching 10 more instances using the AutoScaling schedule.**
    3. During the pre-vacation period setup a scenario where the organization has 15 instances running and 5 instances to scale up and down using Auto Scaling based on the network I/O policy.
    4. During the pre-vacation period setup 20 instances to run continuously.
28. When Auto Scaling is launching a new instance based on condition, which of the below mentioned policies will it follow?
    1. Based on the criteria defined with cross zone Load balancing
    2. Launch an instance which has the highest load distribution
    3. **Launch an instance in the AZ with the fewest instances**
    4. Launch an instance in the AZ which has the highest instances
29. The user has created multiple AutoScaling groups. The user is trying to create a new AS group but it fails. How can the user know that he has reached the AS group limit specified by AutoScaling in that region?
    1. **Run the command: as-describe-account-limits**
    2. Run the command: as-describe-group-limits
    3. Run the command: as-max-account-limits
    4. Run the command: as-list-account-limits
30. A user is trying to save some cost on the AWS services. Which of the below mentioned options will not help him save cost?
    1. Delete the unutilized EBS volumes once the instance is terminated
    2. Delete the Auto Scaling launch configuration after the instances are terminated (Auto Scaling Launch config does not cost anything)
    3. Release the elastic IP if not required once the instance is terminated
    4. Delete the AWS ELB after the instances are terminated
31. To scale up the AWS resources using manual Auto Scaling, which of the below mentioned parameters should the user change?
    1. Maximum capacity
    2. **Desired capacity**
    3. Preferred capacity
    4. Current capacity
32. For AWS Auto Scaling, what is the first transition state an existing instance enters after leaving steady state in Standby mode?
    1. Detaching
    2. Terminating:Wait
    3. **Pending** (You can put any instance that is in an InService state into a Standby state. This enables you to remove the instance from service, troubleshoot or make changes to it, and then put it back into service. Instances in a Standby state continue to be managed by the Auto Scaling group. However, they are not an active part of your application until you put them back into service. Refer [link](http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/AutoScalingGroupLifecycle.html))
    4. EnteringStandby
33. For AWS Auto Scaling, what is the first transition state an instance enters after leaving steady state when scaling in due to health check failure or decreased load?
    1. **Terminating**(When Auto Scaling responds to a scale in event, it terminates one or more instances. These instances are detached from the Auto Scaling group and enter the Terminating state. Refer [link](http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/AutoScalingGroupLifecycle.html))
    2. Detaching
    3. Terminating:Wait
    4. EnteringStandby
34. A user has setup Auto Scaling with ELB on the EC2 instances. The user wants to configure that whenever the CPU utilization is below 10%, Auto Scaling should remove one instance. How can the user configure this?
    1. The user can get an email using SNS when the CPU utilization is less than 10%. The user can use the desired capacity of Auto Scaling to remove the instance
    2. Use CloudWatch to monitor the data and Auto Scaling to remove the instances using scheduled actions
    3. Configure CloudWatch to send a notification to Auto Scaling Launch configuration when the CPU utilization is less than 10% and configure the Auto Scaling policy to remove the instance
    4. **Configure CloudWatch to send a notification to the Auto Scaling group when the CPU Utilization is less than 10% and configure the Auto Scaling policy to remove the instance**
35. A user has enabled detailed CloudWatch metric monitoring on an Auto Scaling group. Which of the below mentioned metrics will help the user identify the total number of instances in an Auto Scaling group including pending, terminating and running instances?
    1. **GroupTotalInstances**(Refer [link](http://docs.aws.amazon.com/autoscaling/latest/userguide/as-instance-monitoring.html#as-group-metrics))
    2. GroupSumInstances
    3. It is not possible to get a count of all the three metrics together. The user has to find the individual number of running, terminating and pending instances and sum it
    4. GroupInstancesCount
36. Your startup wants to implement an order fulfillment process for selling a personalized gadget that needs an average of 3-4 days to produce with some orders taking up to 6 months you expect 10 orders per day on your first day. 1000 orders per day after 6 months and 10,000 orders after 12 months. Orders coming in are checked for consistency then dispatched to your manufacturing plant for production quality control packaging shipment and payment processing. If the product does not meet the quality standards at any stage of the process employees may force the process to repeat a step. Customers are notified via email about order status and any critical issues with their orders such as payment failure. Your case architecture includes AWS Elastic Beanstalk for your website with an RDS MySQL instance for customer data and orders. How can you implement the order fulfillment process while making sure that the emails are delivered reliably? **[PROFESSIONAL]**
    1. Add a business process management application to your Elastic Beanstalk app servers and re-use the ROS database for tracking order status use one of the Elastic Beanstalk instances to send emails to customers.
    2. Use SWF with an Auto Scaling group of activity workers and a decider instance in another Auto Scaling group with min/max=1 Use the decider instance to send emails to customers.
    3. **Use SWF with an Auto Scaling group of activity workers and a decider instance in another Auto Scaling group with min/max=1 use SES to send emails to customers.**
    4. Use an SQS queue to manage all process tasks Use an Auto Scaling group of EC2 Instances that poll the tasks and execute them. Use SES to send emails to customers.
37. Your application is running on EC2 in an Auto Scaling group. Bootstrapping is taking 20 minutes to complete. You find out that instances are shown as InService although the bootstrapping has not completed. How can you make sure that new instances are not added until the bootstrapping has finished. Choose the correct answer:
    1. Create a CloudWatch alarm with an SNS topic to send alarms to your DevOps engineer.
    2. **Create a lifecycle hook to keep the instance in pending:wait state until the bootstrapping has finished and then put the instance in pending:proceed state.**
    3. Increase the number of instances in your Auto Scaling group.
    4. Create a lifecycle hook to keep the instance in standby state until the bootstrapping has finished and then put the instance in pending:proceed state.
38. When a scale out event occurs, the Auto Scaling group launches the required number of EC2 instances using its assigned launch configuration. What instance state do these instances start in? Choose the correct answer:
    1. pending:wait
    2. InService
    3. **Pending**
    4. Terminating
39. With AWS Auto Scaling, once we apply a hook and the action is complete or the default wait state timeout runs out, the state changes to what, depending on which hook we have applied and what the instance is doing? Select two. Choose the 2 correct answers:
    1. **pending:proceed**
    2. pending:wait
    3. terminating:wait
    4. **terminating:proceed**
40. For AWS Auto Scaling, what is the first transition state an existing instance enters after leaving steady state in Standby mode?
    1. Detaching
    2. Terminating:Wait
    3. **Pending** (You can put any instance that is in an InService state into a Standby state. This enables you to remove the instance from service, troubleshoot or make changes to it, and then put it back into service. Instances in a Standby state continue to be managed by the Auto Scaling group. However, they are not an active part of your application until you put them back into service. Refer [link](http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/AutoScalingGroupLifecycle.html))
    4. EnteringStandby
41. For AWS Auto Scaling, what is the first transition state an instance enters after leaving steady state when scaling in due to health check failure or decreased load?
    1. **Terminating**(When Auto Scaling responds to a scale in event, it terminates one or more instances. These instances are detached from the Auto Scaling group and enter the Terminating state. Refer [link](http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/AutoScalingGroupLifecycle.html))
    2. Detaching
    3. Terminating:Wait
    4. EnteringStandby