JON BONSO AND KENNETH SAMONTE

AWS CERTIFIED

DEVOPS ENGINEER PROFESSIONAL

Tutorials Dojo
Study Guide and Cheat Sheets

	to AWS SNS or a Lambda function to send results to Slack channel
Need to run an AWS CodePipeline every day for updating the development progress status	Create CloudWatch Events rule to run on schedule every day and set a target to the AWS CodePipeline ARN
Automate deployment of a Lambda function and test for only 10% of traffic for 10 minutes before allowing 100% traffic flow.	Use CodeDeploy and select deployment configuration CodeDeployDefault.LambdaCanary10Percent10Minutes
Deployment of Elastic Beanstalk application with absolutely no downtime. The solution must maintain full compute capacity during deployment to avoid service degradation.	Choose the “Rolling with additional Batch” deployment policy in Elastic Beanstalk
Deployment of Elastic Beanstalk application where the new version must not be mixed with the current version.	Choose the “Immutable deployments” deployment policy in Elastic Beanstalk
Configuration Management and Infrastructure-as-Code	
The resources on the parent CloudFormation stack needs to be referenced by other nested CloudFormation stacks	Use Export on the Output field of the main CloudFormation stack and use Fn::ImportValue function to import the value on the other stacks
On which part of the CloudFormation template should you define the artifact zip file on the S3 bucket?	The artifact file is defined on the AWS::Lambda::Function code resource block
Need to define the AWS Lambda function inline in the CloudFormation template	On the AWS::Lambda::Function code resource block, the inline function must be enclosed inside the ZipFile section.
Use CloudFormation to update Auto Scaling Group and only terminate the old instances when the newly launched instances become fully operational	Set AutoScalingReplacingUpdate : WillReplace property to TRUE to have CloudFormation retain the old ASG until the instances on the new ASG are healthy.
You need to scale-down the EC2 instances at night when there is low traffic using OpsWorks.	Create *Time-based* instances for automatic scaling of predictable workload.

Can't install an agent on on-premises servers but need to collect information for migration	Deploy the Agentless Discovery Connector VM on your on-premises data center to collect information.
Syntax for CloudFormation with an Amazon ECS cluster with ALB	Use the AWS::ECS::Service element for the ECS Cluster, AWS::ECS::TaskDefinition element for the ECS Task Definitions and the AWS::ElasticLoadBalancingV2::LoadBalancer element for the ALB.
Monitoring and Logging	
Need to centralize audit and collect configuration setting on all regions of multiple accounts	Setup an Aggregator on AWS Config.
Consolidate CloudTrail log files from multiple AWS accounts	Create a central S3 bucket with bucket policy to grant cross-account permission. Set this as destination bucket on the CloudTrail of the other AWS accounts.
Ensure that CloudTrail logs on the S3 bucket are protected and cannot be tampered with.	Enable Log File Validation on CloudTrail settings
Need to collect/investigate application logs from EC2 or on-premises server	Install CloudWatch Logs Agent to send the logs to CloudWatch Logs for storage and viewing.
Need to review logs from running ECS Fargate tasks	Enable awslogs log driver on the Task Definition and add the required logConfiguration parameter.
Need to run real-time analysis for collected application logs	Send logs to CloudWatch Logs, create a Lambda subscription filter, Elasticsearch subscription filter, or Kinesis stream filter.
Need to be automatically notified if you are reaching the limit of running EC2 instances or limit of Auto Scaling Groups	Track service limits with Trusted Advisor on CloudWatch Alarms using the ServiceLimitUsage metric.
Policies and Standards Automation	
Need to secure the buildspec.yml file which contains the AWS keys and database password stored in plaintext.	Store these values as encrypted parameter on SSM Parameter Store

Using default IAM policies for AWSCodeCommitPowerUser but must be limited to a specific repository only	Attach additional policy with Deny rule and custom condition if it does not match the specific repository or branch
You need to secure an S3 bucket by ensuring that only HTTPS requests are allowed for compliance purposes.	Create an S3 bucket policy that Deny if checks for condition aws:SecureTransport is **false**
Need to store a secret, database password, or variable, in the most cost-effective solution	Store the variable on SSM Parameter Store and enable encryption
Need to generate a secret password and have it rotated automatically at regular intervals	Store the secret on AWS Secrets Manager and enable key rotation.
Several team members, with designated roles, need to be granted permission to use AWS resources	Assign AWS managed policies on the IAM accounts such as, ReadOnlyAccess, AdministratorAccess, PowerUserAccess
Apply latest patches on EC2 and automatically create an AMI	Use Systems Manager automation to execute an Automation Document that installs OS patches and creates a new AMI.
Need to have a secure SSH connection to EC2 instances and have a record of all commands executed during the session	Install SSM Agent on EC2 and use SSM Session Manager for the SSH access. Send the session logs to S3 bucket or CloudWatch Logs for auditing and review.
Ensure that the managed EC2 instances have the correct application version and patches installed.	Use SSM Inventory to have a visibility of your managed instances and identify their current configurations.
Apply custom patch baseline from a custom repository and schedule patches to managed instances	Use SSM Patch Manager to define a custom patch baseline and schedule the application patches using SSM Maintenance Windows
Incident and Event Response	
Need to get a notification if somebody deletes files in your S3 bucket	Setup Amazon S3 Event Notifications to get notifications based on specified S3 events on a particular bucket.
Need to be notified when an RDS Multi-AZ failover happens	Setup Amazon RDS Event Notifications to detect specific events on RDS.

Get a notification if somebody uploaded IAM access keys on any public GitHub repositories	Create a CloudWatch Events rule for the AWS_RISK_CREDENTIALS_EXPOSED event from AWS Health Service. Use AWS Step Functions to automatically delete the IAM key.
Get notified on Slack when your EC2 instance is having an AWS-initiated maintenance event	Create a CloudWatch Events rule for the AWS Health Service to detect EC2 Events. Target a Lambda function that will send a notification to the Slack channel
Get notified of any AWS maintenance or events that may impact your EC2 or RDS instances	Create a CloudWatch Events rule for detecting any events on AWS Health Service and send a message to an SNS topic or invoke a Lambda function.
Monitor scaling events of your Amazon EC2 Auto Scaling Group such as launching or terminating an EC2 instance.	Use Amazon EventBridge or CloudWatch Events for monitoring the Auto Scaling Service and monitor the EC2 Instance-Launch Successful and EC2 Instance-Terminate Successful events.
View object-level actions of S3 buckets such as upload or deletion of object in CloudTrail	Set up Data events on your CloudTrail trail to record object-level API activity on your S3 buckets.
Execute a custom action if a specific CodePipeline stage has a FAILED status	Create CloudWatch Event rule to detect failed state on the CodePipeline service, and set a target to SNS topic for notification or invoke a Lambda function to perform custom action.
Automatically rollback a deployment in AWS CodeDeploy when the number of healthy instances is lower than the minimum requirement.	On CodeDeploy, create a deployment alarm that is integrated with Amazon CloudWatch. Track the MinimumHealthyHosts metric for the threshold of EC2 instances and trigger the rollback if the alarm is breached.
Need to complete QA testing before deploying a new version to the production environment	Add a Manual approval step on AWS CodePipeline, and instruct the QA team to approve the step before the pipeline can resume the deployment.
Get notified for OpsWorks auto-healing events	Create a CloudWatch Events rule for the OpsWorks Service to track the auto-healing events

High Availability, Fault Tolerance, and Disaster Recovery	
Need to ensure that both the application and the database are running in the event that one Availability Zone becomes unavailable.	Deploy your application on multiple Availability Zones and set up your Amazon RDS database to use Multi-AZ Deployments.
In the event of an AWS Region outage, you have to make sure that both your application and database will still be running to avoid any service outages.	Create a copy of your deployment on the backup AWS region. Set up an RDS Read-Replica on the backup region.
Automatically switch traffic to the backup region when your primary AWS region fails	Set up Route 53 Failover routing policy with health check enabled on your primary region endpoint.
Need to ensure the availability of a legacy application running on a single EC2 instance	Set up an Auto Scaling Group with MinSize=1 and MaxSize=1 configuration to set a fixed count and ensure that it will be replaced when the instance becomes unhealthy
Ensure that every EC2 instance on an Auto Scaling group downloads the latest code first before being attached to a load balancer	Create an Auto Scaling Lifecycle hook and configure the Pending:Wait hook with the action to download all necessary packages.
Ensure that all EC2 instances on an Auto Scaling group upload all log files in the S3 bucket before being terminated.	Use the Auto Scaling Lifecycle and configure the Terminating:Wait hook with the action to upload all logs to the S3 bucket.

Validate Your Knowledge

After your review, you should take some practice tests to measure your preparedness for the real exam. AWS offers a sample practice test for free which you can find here. You can also opt to buy the longer AWS sample practice test at aws.training, and use the discount coupon you received from any previously taken certification exams. Be aware though that the sample practice tests do not mimic the difficulty of the real DevOps Pro exam.

Therefore, we highly encourage using other mock exams such as our very own **AWS Certified DevOps Engineer Professional Practice Exam** course which contains high-quality questions with complete explanations on correct and incorrect answers, visual images and diagrams, YouTube videos as needed, and also contains reference links to official AWS documentation as well as our cheat sheets and study guides. You can also pair

AutoScalingReplacingUpdate vs AutoScalingRollingUpdate Policy

When using AWS CloudFormation to provision your Auto Scaling groups, you can control how CloudFormation handles updates for your Auto Scaling Group. You need to define the proper **UpdatePolicy** attribute for your ASG depending on your desired behavior during an update.

DevOps Exam Notes:

You can use **AWS::AutoScaling::AutoScalingGroup** resource type on CloudFormation if you want to create an AutoScaling group for your fleet of EC2 instances. Going into the exam, you will need to distinguish the difference between the **AutoScalingReplacingUpdate** and **AutoScalingRollingUpdate** UpdatePolicy attribute, which define how the instances on your group will be updated when you deploy a new revision of your application.

AutoScalingReplacingUpdate - will create a new auto scaling group with new launch configuration. This is more like an immutable type of deployment.

AutoScalingRollingUpdate - will replace the instances on the current auto-scaling group. You can control if instances will be replaced “all-at-once” or use a rolling update by batches. The default behavior is to delete instances first, before creating the new instances.

The **AutoScalingReplacingUpdate** policy specifies how AWS CloudFormation handles replacement updates for an Auto Scaling group. This policy enables you to specify whether AWS CloudFormation replaces an Auto Scaling group with a new one or replaces only the instances in the Auto Scaling group.

```
"UpdatePolicy" : {
  "AutoScalingReplacingUpdate" : {
    "WillReplace" : Boolean
  }
}
```

For example, you can set **AutoScalingReplacingUpdate WillReplace** property to TRUE to have CloudFormation retain the old ASG and the instances it contains. CloudFormation will wait for the successful creation of the new ASG and its instances before it deletes the old ASG. This is helpful when the update fails; CloudFormation can quickly rollback as it will only delete the new ASG. The current ASG and its instances are not affected during the deployment and rollback process.

The **AutoScalingRollingUpdate** policy specifies how AWS CloudFormation handles rolling updates for an Auto Scaling group. Rolling updates enable you to specify whether AWS CloudFormation updates instances that are in an Auto Scaling group in batches or all at once.

```
"UpdatePolicy" : {
  "AutoScalingRollingUpdate" : {
      "MaxBatchSize" : Integer,
      "MinInstancesInService" : Integer,
      "MinSuccessfulInstancesPercent" : Integer,
      "PauseTime" : String,
      "SuspendProcesses" : [ List of processes ],
      "WaitOnResourceSignals" : Boolean
  }
}
```

For example, **AutoScalingRollingUpdate** allows you to specify the **MaxBatchSize** property to set the maximum number of instances that AWS CloudFormation updates at any given time. Or use the **MinInstancesInService** property to ensure that there is a minimum number of instances that are in service while CloudFormation updates the old instances.

Sources:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-updatepolicy.html
https://aws.amazon.com/premiumsupport/knowledge-center/auto-scaling-group-rolling-updates/

Time-Based vs Load-Based Instance

With OpsWorks stacks, you can create a layer of auto scaling EC2 instances and a load balancer to host your application and service traffic from the public Internet. You can set up AWS OpsWorks Stacks to start or stop EC2 instances based on the configuration of your instances. You can use this to automatically increase/decrease the size of your EC2 cluster depending on when you need computing power. This automatic scaling is based on two instance types:

1. **Time-based instances -** allow a stack to handle loads that follow a predictable pattern by including instances that run only at certain times or on certain days. This is helpful in situations when you want to start some instances after 6PM to perform nightly backup tasks or stop some instances on weekends when traffic is lower.

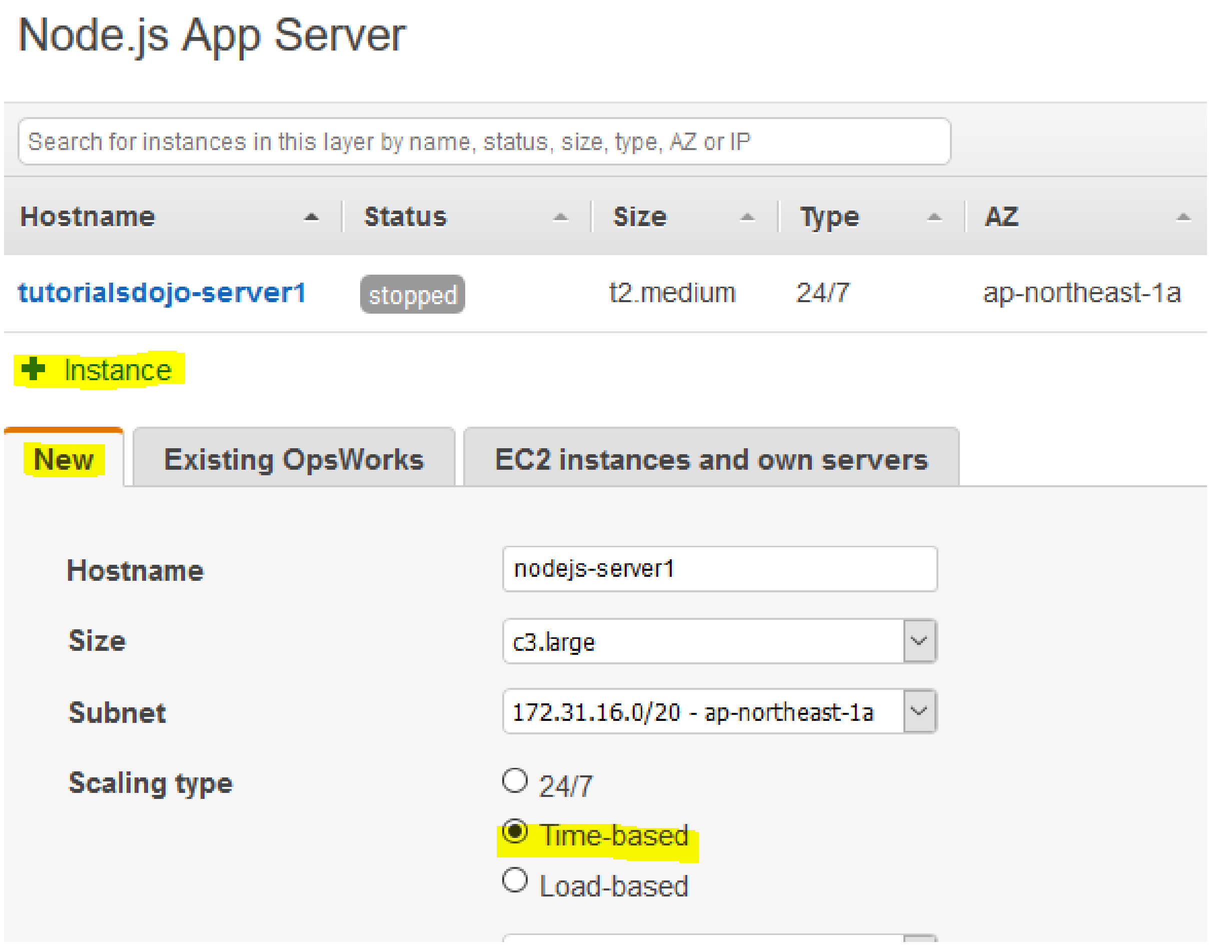
2. **Load-based instances -** allow a stack to handle variable loads by starting additional instances when traffic is high and stopping instances when traffic is low, based on any of several load metrics. This is helpful in situations where you want to start instances when the average CPU utilization exceeds 80% and stop instances when the average CPU load falls below 60%.

A common practice is to use all three instance types together, as follows:

- A set 24/7 instances to handle the base load. You typically just start these instances and let them run continuously.
- A set of time-based instances, which AWS OpsWorks Stacks starts and stops to handle predictable traffic variations.
- A set of load-based instances, which AWS OpsWorks Stacks starts and stops to handle unpredictable traffic variations.

If you want to add a **time-based instance** to a layer follow these steps:

Click the Instances page, click + Instance to add an instance. On the New tab, click Advanced >> and then click time-based.


Node.js App Server

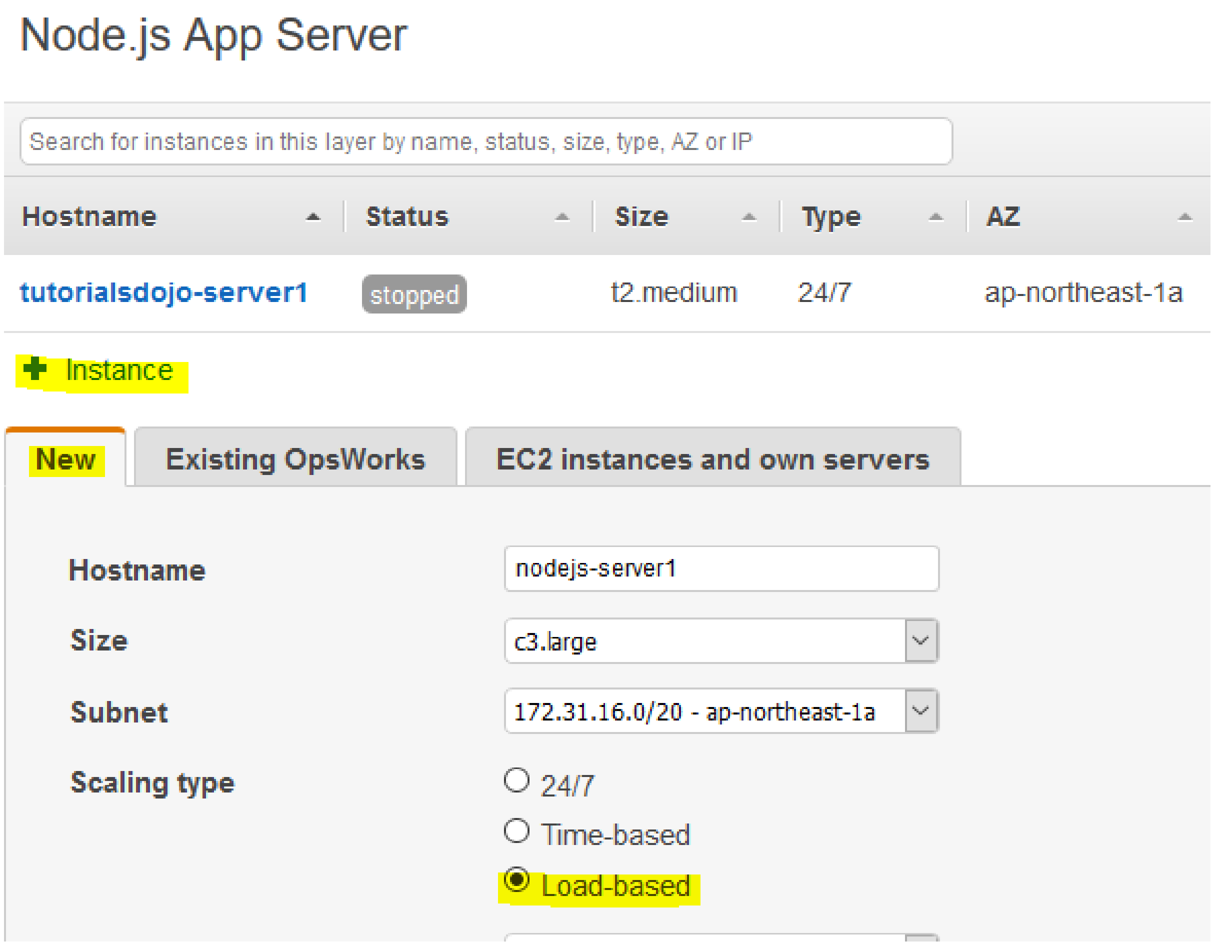
Search for instances in this layer by name, status, size, type, AZ or IP

Hostname	Status	Size	Type	AZ
tutorialsdojo-server1	stopped	t2.medium	24/7	ap-northeast-1a

Hostname	nodejs-server1
Size	c3.large
Subnet	172.31.16.0/20 - ap-northeast-1a
Scaling type	24/7 [✓ Time-based] Load-based

If you want to add a **load-based instance** to a layer:

On the Instances page, click +Instance to add an instance. Click Advanced >> and then click load-based.



Node.js App Server

Search for instances in this layer by name, status, size, type, AZ or IP

Hostname	Status	Size	Type	AZ
tutorialsdojo-server1	stopped	t2.medium	24/7	ap-northeast-1a

Hostname	nodejs-server1
Size	c3.large
Subnet	172.31.16.0/20 - ap-northeast-1a
Scaling type	○ 24/7 ○ Time-based [✓ Load-based]

Source:
https://docs.aws.amazon.com/opsworks/latest/userguide/workinginstances-autoscaling.html

CloudFormation Template for ECS, Auto Scaling and ALB

Amazon Elastic Container Service (ECS) allows you to manage and run Docker containers on clusters of EC2 instances. You can also configure your ECS to use Fargate launch type which eliminates the need to manage EC2 instances.

With CloudFormation, you can define your ECS clusters and tasks definitions to easily deploy your containers. For high availability of your Docker containers, ECS clusters are usually configured with an auto scaling group behind an application load balancer. These resources can also be declared on your CloudFormation template.

DevOps Exam Notes:

Going on to the exam, be sure to remember the syntax needed to declare your ECS cluster, Auto Scaling group, and application load balancer. The **AWS::ECS::Service** resource creates an ECS cluster and the **AWS::ECS::TaskDefinition** resource creates a task definition for your container. The **AWS::ElasticLoadBalancingV2::LoadBalance**r resource creates an application load balancer and the **AWS::AutoScaling::AutoScalingGroup** resource creates an EC2 auto scaling group.

AWS provides an example template which you can use to deploy a web application in an Amazon ECS container with auto scaling and application load balancer. Here's a snippet of the template with the core resources:

```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Resources":{
      "ECSCluster":{
      "Type":"AWS::ECS::Cluster"

      },

......

      "taskdefinition":{
      "Type":"AWS::ECS::TaskDefinition",

      "Properties":{

......

      "ECSALB":{
      "Type":"AWS::ElasticLoadBalancingV2::LoadBalancer",
      "Properties":{

......

      "ECSAutoScalingGroup":{
      "Type":"AWS::AutoScaling::AutoScalingGroup",
      "Properties":{
      "VPCZoneIdentifier":{
            "Ref":"SubnetId"

      },
```

Sources:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/quickref-ecs.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ecs-service.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ecs-service-loadbalancers.html

Monitoring Service Limits with Trusted Advisor

AWS Trusted Advisor provides you real-time guidance to help you provision your resources following AWS best practices. It provides recommendations on 5 categories – Cost Optimization, Performance, Security, Fault Tolerance and Service Limits.

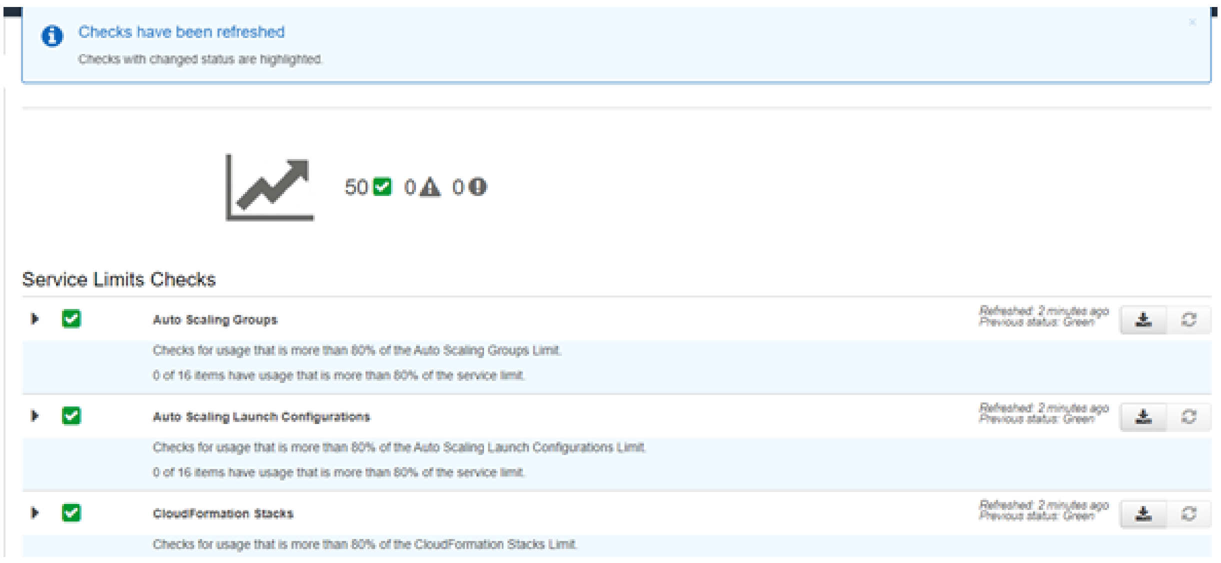
With Trusted Advisor's Service Limit Dashboard, you can view, refresh, and export utilization and limit data on a per-limit basis. These metrics are published on AWS CloudWatch in which you can create custom alarms based on percentage of service utilization against limits, understand the number of resources by each check, or view time-aggregate views of check results across service categories.

DevOps Exam Notes:

You need to understand that service limits are important when managing your AWS resources. In the exam you can be given a scenario in which you have several Auto Scaling groups in your AWS account and you need to make sure that you are not reaching the service limit when you perform your blue/green deployments for your application. You can track service limits with Trusted Advisor and CloudWatch Alarms. The ServiceLimitUsage metric on CloudWatch Alarms is only visible for Business and Enterprise support customers.

Here's how you can create a CloudWatch Alarm to detect if you are nearing your auto scaling service limit and send a notification so you can request for a service limit increase to AWS support.

1. First, head over to AWS Trusted Advisor > Service Limits and click the refresh button. This will refresh the service limit status for your account.



Checks have been refreshed
Checks with changed status are highlighted.

50 ✓ 0 ⚠ 0 !

Service Limits Checks

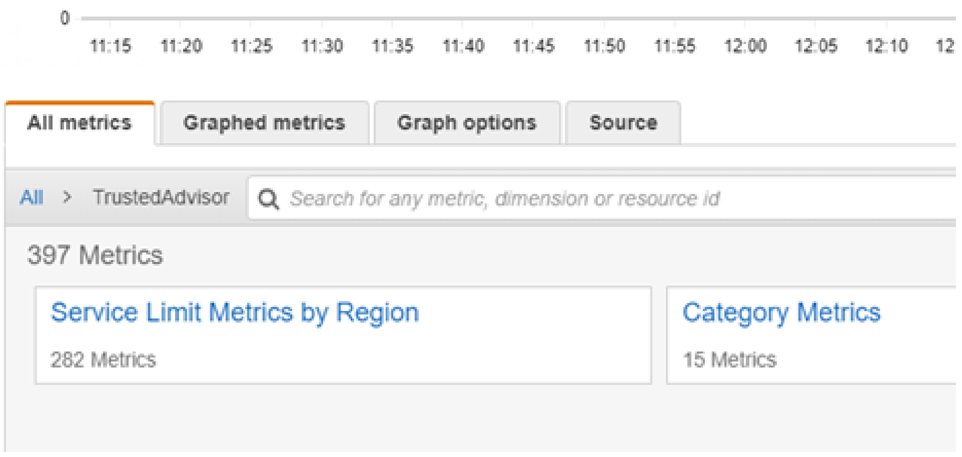
Auto Scaling Groups — Refreshed: 2 minutes ago, Previous status: Green
Checks for usage that is more than 80% of the Auto Scaling Groups Limit.
0 of 16 items have usage that is more than 80% of the service limit.

Auto Scaling Launch Configurations — Refreshed: 2 minutes ago, Previous status: Green
Checks for usage that is more than 80% of the Auto Scaling Launch Configurations Limit.
0 of 16 items have usage that is more than 80% of the service limit.

CloudFormation Stacks — Refreshed: 2 minutes ago, Previous status: Green
Checks for usage that is more than 80% of the CloudFormation Stacks Limit.

2. Go to CloudWatch > Alarms. Make sure that you are in the North Virginia region. Click the “Create Alarm” button and click “Select Metric”.

3. In the “All metrics” tab, click “Trusted Advisor” category and you will see “Service Limits by Region”.

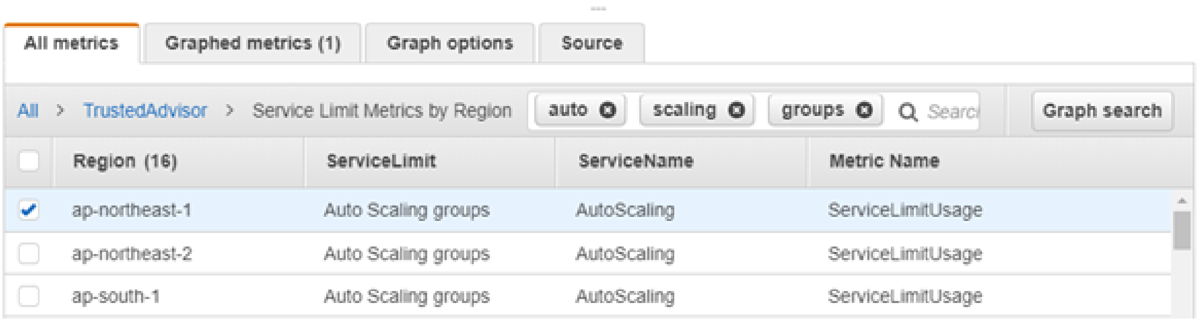


397 Metrics

Service Limit Metrics by Region
282 Metrics

Category Metrics
15 Metrics

4. Search for Auto Scaling groups on your desired region and click Select Metric.

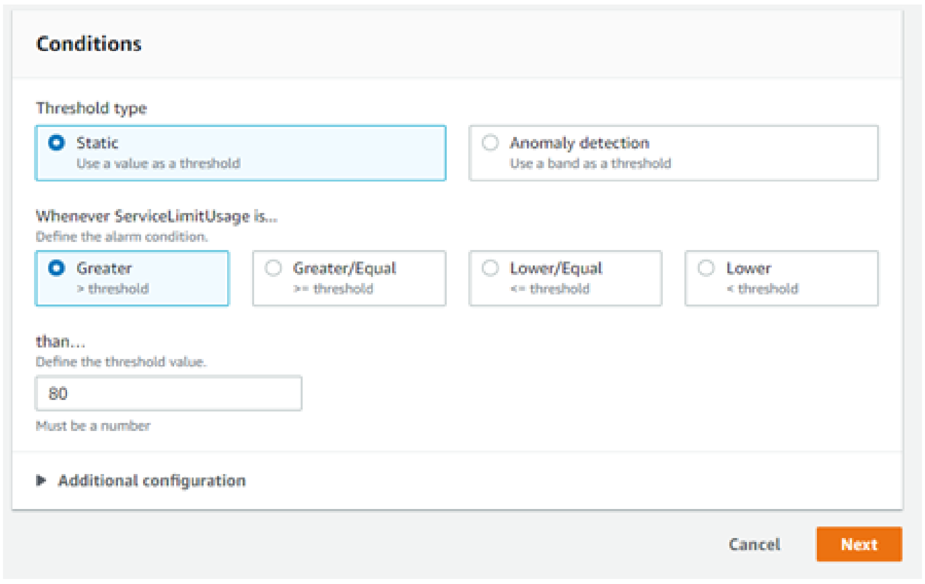


All metrics | Graphed metrics (1) | Graph options | Source

All > TrustedAdvisor > Service Limit Metrics by Region — auto, scaling, groups — Graph search

	Region (16)	ServiceLimit	ServiceName	Metric Name
✓	ap-northeast-1	Auto Scaling groups	AutoScaling	ServiceLimitUsage
	ap-northeast-2	Auto Scaling groups	AutoScaling	ServiceLimitUsage
	ap-south-1	Auto Scaling groups	AutoScaling	ServiceLimitUsage

5. We can set the condition for this Alarm so that when your Auto Scaling group reaches 80 for that particular region, the alarm is triggered.



Conditions

Threshold type
[✓ Static — Use a value as a threshold] Anomaly detection — Use a band as a threshold

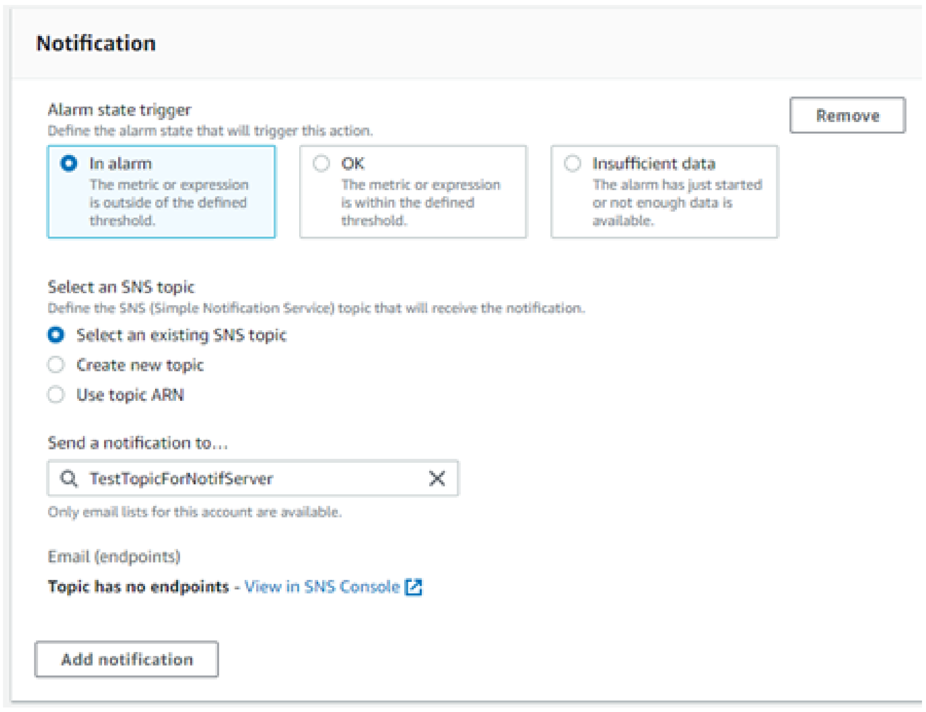
Whenever ServiceLimitUsage is...
Define the alarm condition.
[✓ Greater — > threshold] Greater/Equal — >= threshold Lower/Equal — <= threshold Lower — < threshold

than...
Define the threshold value.
80
Must be a number

▶ Additional configuration

Cancel | Next

6. You can then configure an SNS topic to receive a notification for this alarm.



Notification

Alarm state trigger
Define the alarm state that will trigger this action.

[✓ In alarm] The metric or expression is outside of the defined threshold. | OK The metric or expression is within the defined threshold. | Insufficient data The alarm has just started or not enough data is available.

Select an SNS topic
Define the SNS (Simple Notification Service) topic that will receive the notification.

[✓ Select an existing SNS topic] Create new topic | Use topic ARN

Send a notification to…
TestTopicForNotifServer
Only email lists for this account are available.

Email (endpoints)
Topic has no endpoints - View in SNS Console

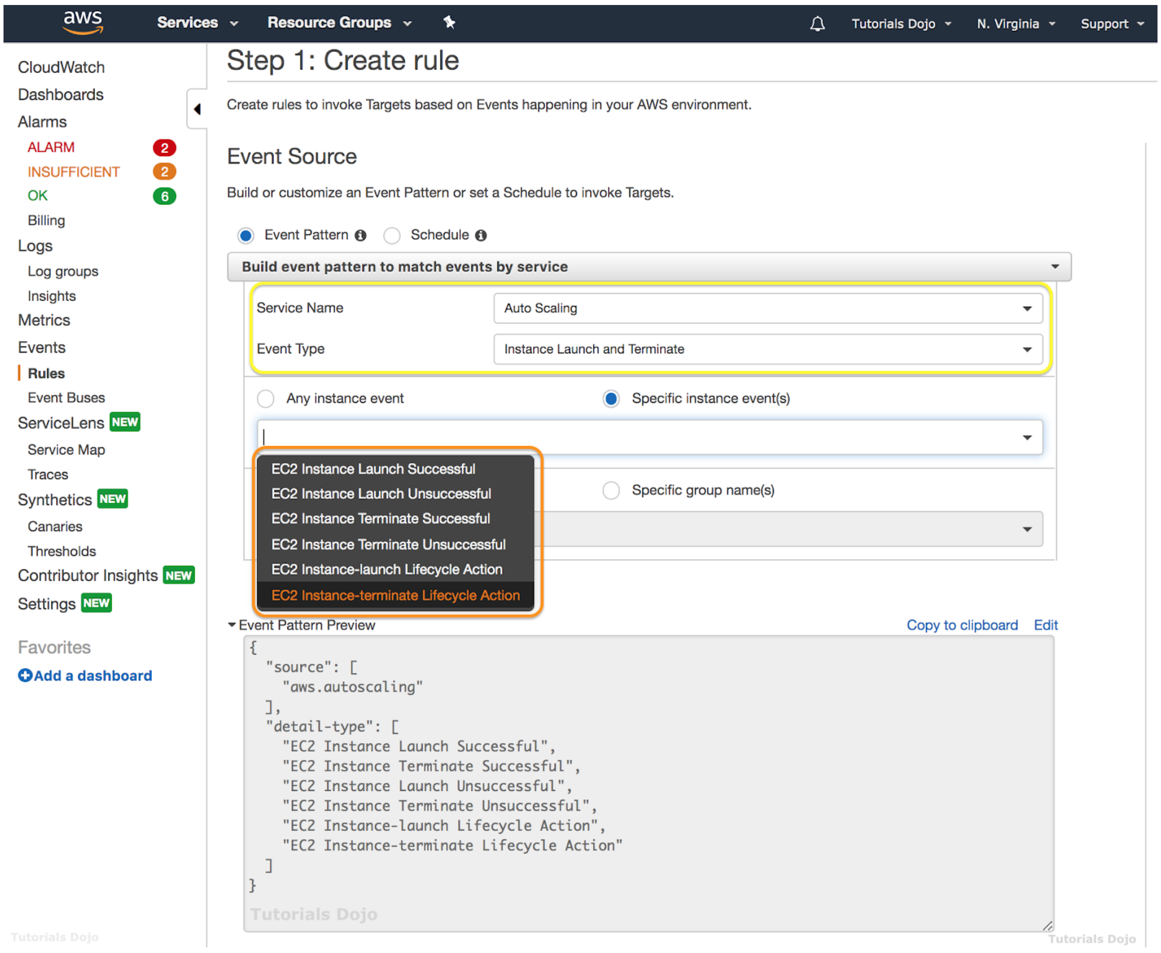
7. Click Next to Preview the alarm and click “Create Alarm” to create the alarm.

Sources:
https://aws.amazon.com/about-aws/whats-new/2017/11/aws-trusted-advisor-adds-service-limit-dashboard-and-cloudwatch-metrics/
https://aws.amazon.com/blogs/mt/monitoring-service-limits-with-trusted-advisor-and-amazon-cloudwatch/

Monitoring Amazon EC2 Auto Scaling Events

Auto Scaling provides fault tolerance, high availability, and cost management to your computing resources. It automatically scales in (terminates existing EC2 instances) if the demand is low and scales out (launch new EC2 instances) if the incoming traffic exceeds the specified threshold. AWS offers various ways of monitoring your fleet of Amazon EC2 instances as well responding to Auto Scaling events.

You can either use Amazon EventBridge or Amazon CloudWatch Events to track the Auto Scaling Events and run a corresponding custom action using AWS Lambda. Amazon EventBridge extends the capabilities of Amazon CloudWatch Events to integrate internal and external services. It allows you to track the changes of your Auto Scaling group in near real-time, including your custom applications, Software-as-a-Service (SaaS) partner apps, and other AWS services.



Step 1: Create rule

Create rules to invoke Targets based on Events happening in your AWS environment.

Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

[✓ Event Pattern] Schedule

Build event pattern to match events by service

Service Name: Auto Scaling

Event Type: Instance Launch and Terminate

Any instance event [✓ Specific instance event(s)]

Specific group name(s)

- EC2 Instance Launch Successful
- EC2 Instance Launch Unsuccessful
- EC2 Instance Terminate Successful
- EC2 Instance Terminate Unsuccessful
- EC2 Instance-launch Lifecycle Action
- EC2 Instance-terminate Lifecycle Action

Event Pattern Preview — Copy to clipboard Edit

```
{
  "source": [
    "aws.autoscaling"
  ],
  "detail-type": [
    "EC2 Instance Launch Successful",
    "EC2 Instance Terminate Successful",
    "EC2 Instance Launch Unsuccessful",
    "EC2 Instance Terminate Unsuccessful",
    "EC2 Instance-launch Lifecycle Action",
    "EC2 Instance-terminate Lifecycle Action"
  ]
}
```

Amazon EventBridge or Amazon CloudWatch Events can be used to send events to the specified target when the following events occur:

- EC2 Instance-launch Lifecycle Action
- EC2 Instance Launch Successful
- EC2 Instance Launch Unsuccessful
- EC2 Instance-terminate Lifecycle Action
- EC2 Instance Terminate Successful
- EC2 Instance Terminate Unsuccessful

The **`EC2 Instance-launch Lifecycle Action`** is a scale-out event in which the Amazon EC2 Auto Scaling moved an EC2 instance to a **`Pending:Wait`** state due to a lifecycle hook. Conversely, the **`EC2 Instance-terminate Lifecycle Action`** is a scale-in event in which EC2 Auto Scaling updates an instance to a **`Terminating:Wait`** state.

Source:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/cloud-watch-events.html

Auto Scaling Group with MinSize = 1 and MaxSize = 1

With Auto Scaling, you can set the number of EC2 instances that you need depending on the traffic of your application. However, there will be scenarios on the exam where you will need to set a fixed number of instances.

For example, you have a legacy application hosted on an EC2 instance. The application does not support working on a cluster of EC2 instances so it needs to run on a single EC2 instance and you need to make sure that the application is available and will heal itself even if that EC2 instance crashes.

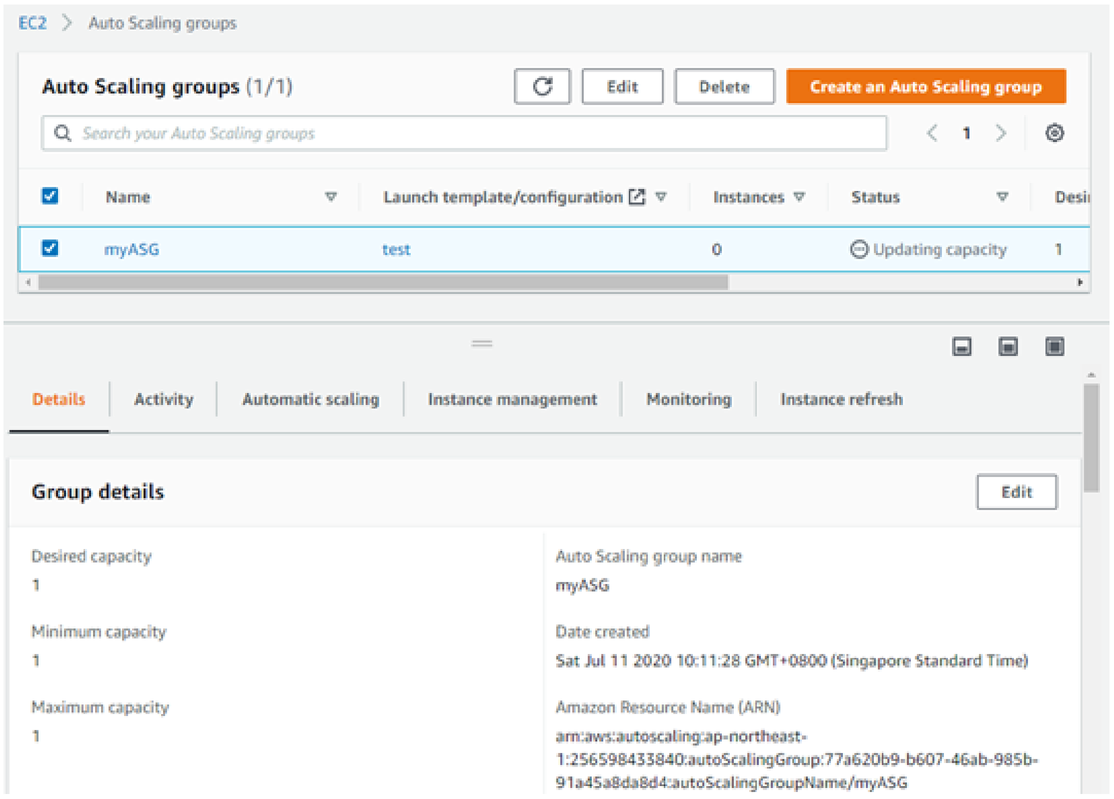
In this scenario, you will create an Auto Scaling Group using the EC2 AMI in the Launch configuration and set the size of the group to Min 1 and Max 1. This ensures that only instances of the application are running. Auto Scaling will perform health checks for the EC2 instances periodically. If the EC2 instance fails the health check, Auto Scaling will replace the instance.

Hence, it will always be available and self-healing. This makes your application fault-tolerant.

To set the MinSize and MaxSize of the Auto Scaling group:

1. Go to the EC2 Console page, on the left pane, choose Auto Scaling Groups.

2. Select the check box next to your Auto Scaling group. The bottom pane will show information on the selected Auto Scaling group.


EC2 > Auto Scaling groups
Auto Scaling groups (1/1)
Edit
Delete
Create an Auto Scaling group
Search your Auto Scaling groups
Details
Activity
Automatic scaling
Instance management
Monitoring
Instance refresh
Edit

Name	Launch template/configuration	Instances	Status	Desi
myASG	test	0	Updating capacity	1

Group details

Desired capacity
1

Minimum capacity
1

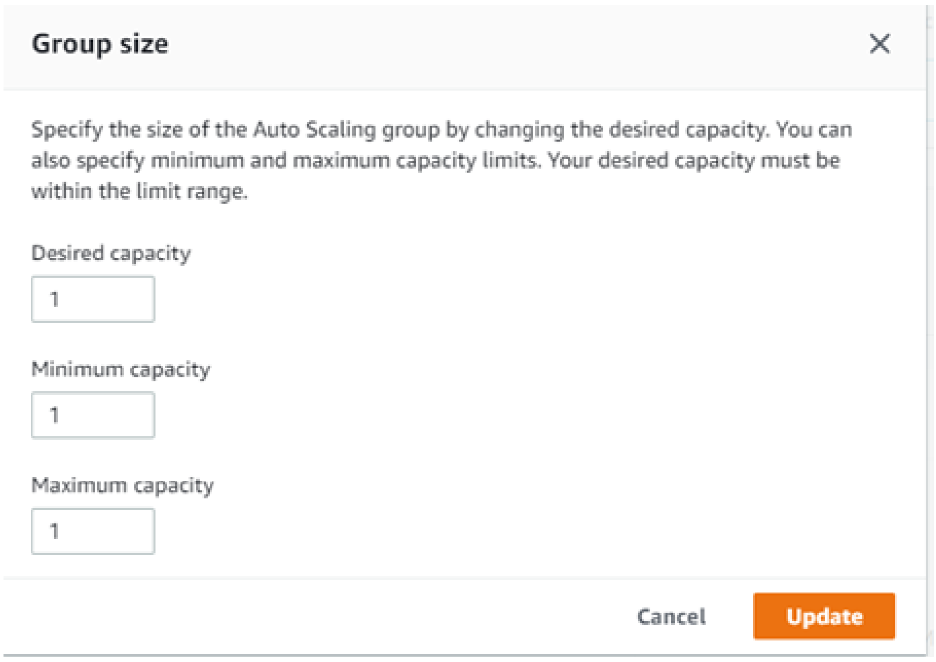
Maximum capacity
1

Auto Scaling group name
myASG

Date created
Sat Jul 11 2020 10:11:28 GMT+0800 (Singapore Standard Time)

Amazon Resource Name (ARN)
arn:aws:autoscaling:ap-northeast-1:256598433840:autoScalingGroup:77a620b9-b607-46ab-985b-91a45a8da8d4:autoScalingGroupName/myASG

3. On the Details tab, view or change the current settings for minimum, maximum, and desired capacity. Set the Desired capacity to 1, Minimum capacity to 1 and Maximum capacity to 1. Make sure that you don’t have any Automatic Scaling policies configured for this group.


Group size

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity: 1

Minimum capacity: 1

Maximum capacity: 1

Sources:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-maintain-instance-levels.html
https://docs.aws.amazon.com/autoscaling/ec2/userguide/asg-capacity-limits.html

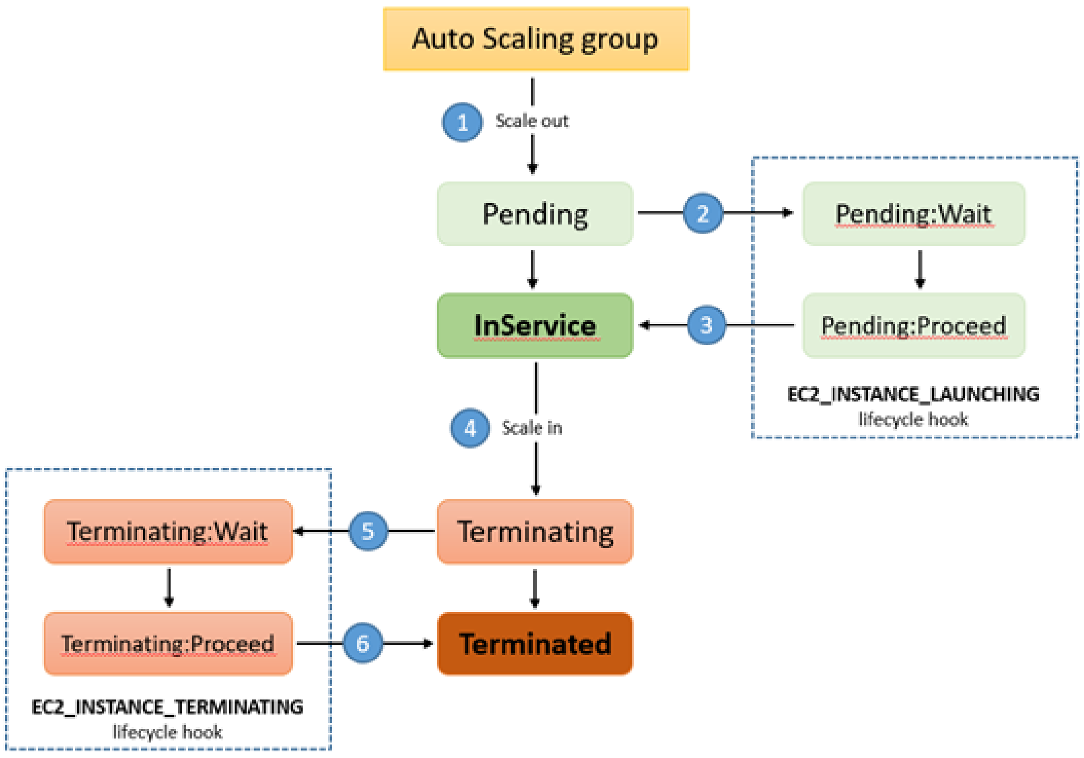
Auto Scaling Lifecycle Hooks

As your Auto Scaling group scales out or scales in your EC2 instances, you may want to perform custom actions before they start accepting traffic or before they get terminated. Auto Scaling Lifecycle Hooks allows you to perform custom actions during these stages. A lifecycle hook puts your EC2 instance into a wait state (**`Pending:Wait`** or **`Terminating:Wait`**) until your custom action has been performed or when the timeout period ends. The EC2 instance stays in a wait state for one hour by default, and then the Auto Scaling group resumes the launch or terminate process (**`Pending:Proceed`** or **`Terminating:Proceed`**).

For example, during the scale-out event of your ASG, you want to make sure that new EC2 instances download the latest code base from the repository and that your EC2 user data has completed before it starts accepting traffic. You can use the Pending:Wait hook. This way, the new instances will be fully ready and will quickly pass the load balancer health check when they are added as targets.

Another example: during the scale-in event of your ASG, suppose your instances upload data logs to S3 every minute. You may want to pause the instance termination for a certain amount of time to allow the EC2 to upload all data logs before it gets completely terminated.

The following diagram shows the transitions between the EC2 instance states with lifecycle hooks.



DevOps Exam Notes:

During the paused state (either launch or terminate), you can do more than just run custom scripts or wait for timeouts. CloudWatch Events receives the scaling action and you can define a CloudWatch Events Target to invoke a Lambda function that can perform custom actions, have it send a notification to your email via SNS, or trigger an SSM Run Command or SSM Automation to perform specific EC2 related tasks.

Sources:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/lifecycle-hooks.html
https://docs.aws.amazon.com/cli/latest/reference/autoscaling/put-lifecycle-hook.html

AWS Auto Scaling

- Configure automatic scaling for the AWS resources quickly through a scaling plan that uses dynamic scaling and predictive scaling.
- Optimize for availability, for cost, or a balance of both.
- Scaling in means decreasing the size of a group while scaling out means increasing the size of a group.
- Useful for:
  - Cyclical traffic such as high use of resources during regular business hours and low use of resources overnight
  - On and off traffic patterns, such as batch processing, testing, or periodic analysis
  - Variable traffic patterns, such as software for marketing campaigns with periods of spiky growth
- **Features**
  - Launch or terminate EC2 instances in an Auto Scaling group.
  - Launch or terminate instances from an EC2 Spot Fleet request, or automatically replace instances that get interrupted for price or capacity reasons.
  - Adjust the ECS service desired count up or down in response to load variations.
  - Use Dynamic Scaling to add and remove capacity for resources to maintain resource utilization at the specified target value.
  - Use Predictive Scaling to forecast your future load demands by analyzing your historical records for a metric. It also allows you to schedule scaling actions that proactively add and remove resource capacity to reflect the load forecast, and control maximum capacity behavior. Only available for EC2 Auto Scaling groups.
  - You can suspend and resume any of your AWS Application Auto Scaling actions.
- **Amazon EC2 Auto Scaling**
  - Ensuring you have the correct number of EC2 instances available to handle your application load using Auto Scaling Groups.
  - An Auto Scaling group 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.
  - You specify the minimum, maximum and desired number of instances in each Auto Scaling group.
  - Key Components

Groups	Your EC2 instances are organized into groups so that they are treated as a logical unit for scaling and management. When you create a group, you can specify its minimum, maximum, and desired number of EC2 instances.

Launch configurations	Your group uses a launch configuration as a template for its EC2 instances. When you create a launch configuration, you can specify information such as the AMI ID, instance type, key pair, security groups, and block device mapping for your instances.
Scaling options	How to scale your Auto Scaling groups.

- Scaling Options
    - Scale to maintain current instance levels at all times
    - Manual Scaling
    - Scale based on a schedule
    - Scale based on a demand
- The cooldown period is a configurable setting that helps ensure to not launch or terminate additional instances before previous scaling activities take effect.
    - EC2 Auto Scaling supports cooldown periods when using simple scaling policies, but not when using target tracking policies, step scaling policies, or scheduled scaling.
- Amazon EC2 Auto Scaling marks an instance as unhealthy if the instance is in a state other than running, the system status is impaired, or Elastic Load Balancing reports that the instance failed the health checks.
- Termination of Instances
    - When you configure automatic scale in, you must decide which instances should terminate first and set up a termination policy. You can also use instance protection to prevent specific instances from being terminated during automatic scale in.
    - Default Termination Policy

    - Custom Termination Policies
        - OldestInstance - Terminate the oldest instance in the group.
        - NewestInstance - Terminate the newest instance in the group.
        - OldestLaunchConfiguration - Terminate instances that have the oldest launch configuration.
        - ClosestToNextInstanceHour - Terminate instances that are closest to the next billing hour.

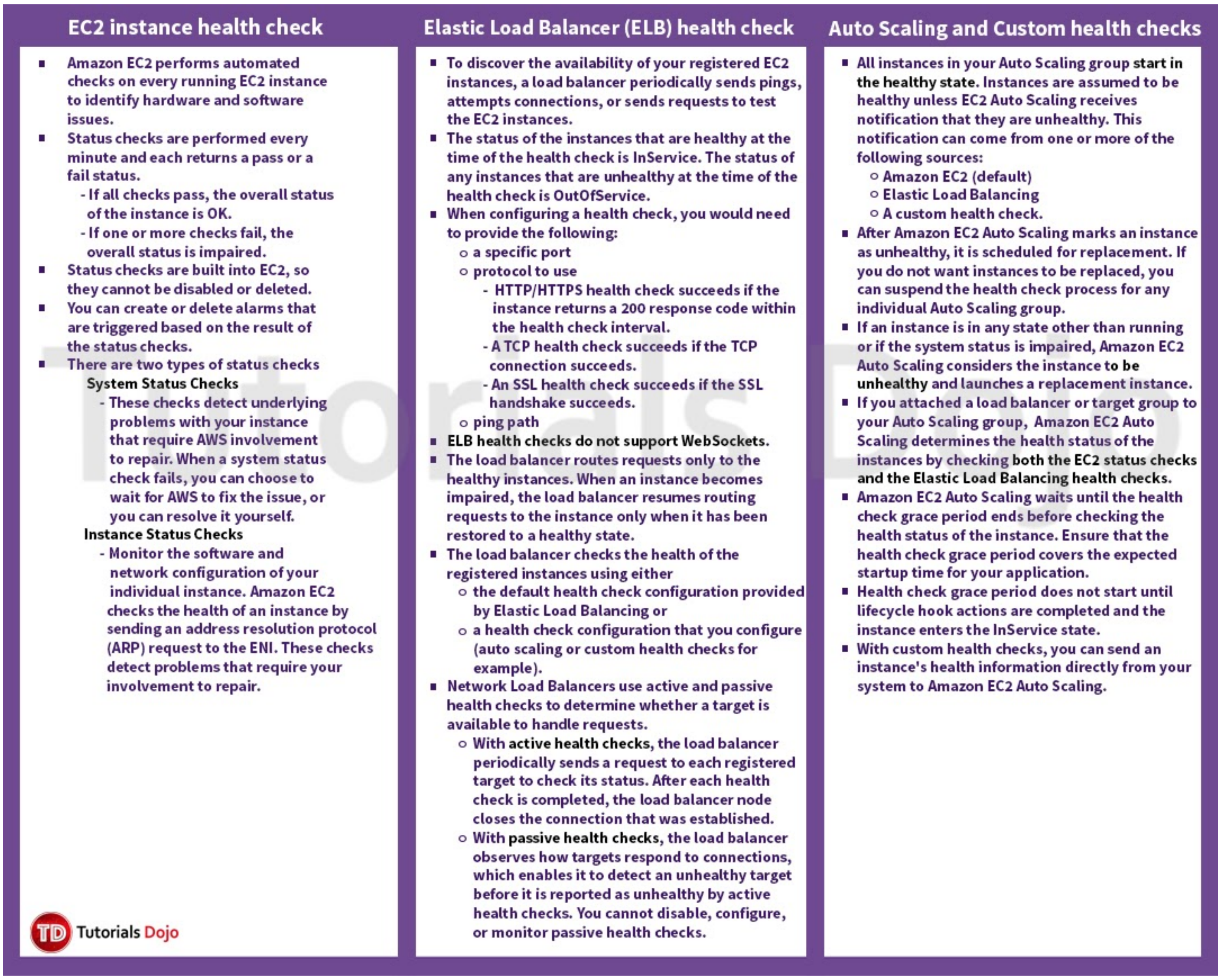
You can create launch templates that specifies instance configuration information when you launch EC2 instances, and allows you to have multiple versions of a template.
A launch configuration is an instance configuration template that an Auto Scaling group uses to launch EC2 instances, and you specify information for the instances.

- You can specify your launch configuration with multiple Auto Scaling groups.
- You can only specify one launch configuration for an Auto Scaling group at a time, and you can't modify a launch configuration after you've created it.

- **Monitoring**
    - Health checks - identifies any instances that are unhealthy

        - Amazon EC2 status checks (default)
        - Elastic Load Balancing health checks
        - Custom health checks.
    - Auto scaling does not perform health checks on instances in the standby state. Standby state can be used for performing updates/changes/troubleshooting without health checks being performed or replacement instances being launched.
    - CloudWatch metrics - enables you to retrieve statistics about Auto Scaling-published data points as an ordered set of time-series data, known as metrics. You can use these metrics to verify that your system is performing as expected.
    - CloudWatch Events - Auto Scaling can submit events to CloudWatch Events when your Auto Scaling groups launch or terminate instances, or when a lifecycle action occurs.
    - SNS notifications - Auto Scaling can send Amazon SNS notifications when your Auto Scaling groups launch or terminate instances.
    - CloudTrail logs - enables you to keep track of the calls made to the Auto Scaling API by or on behalf of your AWS account, and stores the information in log files in an S3 bucket that you specify.

EC2 Instance Health Check vs ELB Health Check vs Auto Scaling and Custom Health Check



EC2 instance health check

- Amazon EC2 performs automated checks on every running EC2 instance to identify hardware and software issues.
- Status checks are performed every minute and each returns a pass or a fail status.
  - If all checks pass, the overall status of the instance is OK.
  - If one or more checks fail, the overall status is impaired.
- Status checks are built into EC2, so they cannot be disabled or deleted.
- You can create or delete alarms that are triggered based on the result of the status checks.
- There are two types of status checks
  - System Status Checks
    - These checks detect underlying problems with your instance that require AWS involvement to repair. When a system status check fails, you can choose to wait for AWS to fix the issue, or you can resolve it yourself.
  - Instance Status Checks
    - Monitor the software and network configuration of your individual instance. Amazon EC2 checks the health of an instance by sending an address resolution protocol (ARP) request to the ENI. These checks detect problems that require your involvement to repair.

Elastic Load Balancer (ELB) health check

- To discover the availability of your registered EC2 instances, a load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances.
- The status of the instances that are healthy at the time of the health check is InService. The status of any instances that are unhealthy at the time of the health check is OutOfService.
- When configuring a health check, you would need to provide the following:
  - a specific port
  - protocol to use
    - HTTP/HTTPS health check succeeds if the instance returns a 200 response code within the health check interval.
    - A TCP health check succeeds if the TCP connection succeeds.
    - An SSL health check succeeds if the SSL handshake succeeds.
  - ping path
- ELB health checks do not support WebSockets.
- The load balancer routes requests only to the healthy instances. When an instance becomes impaired, the load balancer resumes routing requests to the instance only when it has been restored to a healthy state.
- The load balancer checks the health of the registered instances using either
  - the default health check configuration provided by Elastic Load Balancing or
  - a health check configuration that you configure (auto scaling or custom health checks for example).
- Network Load Balancers use active and passive health checks to determine whether a target is available to handle requests.
  - With active health checks, the load balancer periodically sends a request to each registered target to check its status. After each health check is completed, the load balancer node closes the connection that was established.
  - With passive health checks, the load balancer observes how targets respond to connections, which enables it to detect an unhealthy target before it is reported as unhealthy by active health checks. You cannot disable, configure, or monitor passive health checks.

Auto Scaling and Custom health checks

- All instances in your Auto Scaling group start in the healthy state. Instances are assumed to be healthy unless EC2 Auto Scaling receives notification that they are unhealthy. This notification can come from one or more of the following sources:
  - Amazon EC2 (default)
  - Elastic Load Balancing
  - A custom health check.
- After Amazon EC2 Auto Scaling marks an instance as unhealthy, it is scheduled for replacement. If you do not want instances to be replaced, you can suspend the health check process for any individual Auto Scaling group.
- If an instance is in any state other than running or if the system status is impaired, Amazon EC2 Auto Scaling considers the instance to be unhealthy and launches a replacement instance.
- If you attached a load balancer or target group to your Auto Scaling group, Amazon EC2 Auto Scaling determines the health status of the instances by checking both the EC2 status checks and the Elastic Load Balancing health checks.
- Amazon EC2 Auto Scaling waits until the health check grace period ends before checking the health status of the instance. Ensure that the health check grace period covers the expected startup time for your application.
- Health check grace period does not start until lifecycle hook actions are completed and the instance enters the InService state.
- With custom health checks, you can send an instance's health information directly from your system to Amazon EC2 Auto Scaling.