AWS/DEVOPS : CLOUD ENGINEER : 2023

-------------------------------------------------------------------------------------- **Rohan Sen BTech – CSE**

**Mail –** [**rohan77x@gmail.com**](mailto:rohan77x@gmail.com)

**Date : 19/4/2023**

**Compute Services:**

* **Amazon EC2** (Elastic Compute Cloud): Virtual machines (VMs) in the cloud that allow you to run applications.
* **Amazon EC2** **Auto Scaling:** Automatically adjusts the number of EC2 instances based on traffic and demand.
* **Amazon Lightsail**: Simplified compute for small-scale applications and websites.
* **AWS Lambda**: Serverless compute service that allows you to run code in response to events without provisioning or managing servers.

**Storage Services:**

* **Amazon S3** (Simple Storage Service): Scalable object storage for storing and retrieving files, images, videos, and other data.3
* **Amazon EBS** (Elastic Block Store): Persistent block-level storage volumes for EC2 instances.
* **Amazon Glacier**: Long-term storage for archival data.
* **Amazon S3 Transfer Acceleration**: Accelerated transfer of data to Amazon S3 over the public internet.

**Database Services:**

* **Amazon RDS** (Relational Database Service): Managed database service that supports popular relational databases such as MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.
* **Amazon DynamoDB**: Managed NoSQL database for storing key-value and document data.
* **Amazon Aurora**: High-performance managed relational database engine compatible with MySQL and PostgreSQL.
* **Amazon Redshift**: Fully managed data warehouse for big data analytics.

**Networking Services:**

* **Amazon VPC** (Virtual Private Cloud): Isolated virtual network for running resources within a defined network space.
* **Amazon Route 53**: Scalable domain name system (DNS) web service for managing domain names and routing internet traffic.
* **Amazon CloudFront**: Content delivery network (CDN) for delivering content with low latency and high transfer speeds.
* **Amazon API Gateway**: Fully managed service for creating, deploying, and managing APIs (Application Programming Interfaces).
* Management & Monitoring Services:
* **AWS CloudFormation**: Infrastructure as code (IaC) service for provisioning and managing AWS resources using templates.
* **AWS CloudWatch**: Monitoring and observability service for collecting and analyzing metrics, logs, and events.
* **AWS Identity and Access Management (IAM**): Service for managing access and permissions to AWS resources.
* **AWS CloudTrail**: Service that records and monitors AWS API activity for auditing and compliance purposes.

**DevOps Services:**

* **AWS CodeCommit**: Fully managed source control service that hosts secure Git repositories.
* **AWS CodeBuild**: Fully managed build service for compiling source code, running tests, and creating software artifacts.
* **AWS CodeDeploy**: Automated deployment service for deploying applications to EC2 instances or on-premises servers.
* **AWS CodePipeline**: Fully managed continuous delivery service for orchestrating and automating the software release process.

**Compute Services:**

**AMAZON EC2 :**

Amazon EC2, which stands for Amazon Elastic Compute Cloud, is a web service offered by Amazon Web Services (AWS) that provides resizable virtual computing resources in the cloud. It allows users to rent virtual machines (known as "instances") on a pay-as-you-go basis, where they can run applications, store data, and perform various computing tasks without having to invest in physical hardware.

With Amazon EC2, users can choose from a wide variety of instance types, which differ in terms of compute power, memory, storage, and other specifications, to meet their specific requirements. They can also select different operating systems, pre-configured software, and networking options to customize their instances.

Amazon EC2 provides a scalable and flexible solution, allowing users to quickly spin up or shut down instances as needed, and it can be used for a wide range of use cases, such as hosting websites and web applications, running databases, performing data processing, running machine learning models, and more. It also integrates with other AWS services, allowing users to build complex and scalable cloud-based applications.

**Amazon EC2 Auto Scaling:**

Auto Scaling in Amazon Web Services (AWS) is a service that allows you to automatically adjust the number of EC2 instances in an application's fleet based on certain conditions or policies. With Auto Scaling, you can dynamically scale your EC2 instances up or down in response to changes in demand, ensuring that your application has the right amount of resources to handle the workload efficiently.

Auto Scaling provides the ability to define scaling policies, which are rules that determine when and how to add or remove EC2 instances from an Auto Scaling group. An Auto Scaling group is a logical grouping of EC2 instances that share the same configuration and scaling policies. You can specify the minimum and maximum number of instances to maintain in the group, as well as the desired number of instances, which is the target number of instances that should be running at all times.

Auto Scaling monitors the health of instances in the group and automatically replaces failed instances or terminates instances that are no longer needed, based on the defined scaling policies. It also allows you to scale instances proactively based on scheduled events, such as anticipated traffic spikes during peak hours, or based on custom metrics that you define.

Some key features of Auto Scaling include:

1. Automatic scaling: Auto Scaling automatically adds or removes EC2 instances based on the configured scaling policies, ensuring that the right amount of resources are available to handle the workload efficiently.
2. High availability and fault tolerance: Auto Scaling automatically distributes instances across multiple Availability Zones (AZs) to ensure high availability and fault tolerance. It can also detect AZ failures and automatically replace affected instances in healthy AZs.
3. Cost optimization: Auto Scaling helps to optimize costs by automatically scaling instances based on demand, so you only pay for the resources you need at any given time.
4. Flexibility and elasticity: Auto Scaling provides the flexibility to scale instances up or down based on demand, allowing you to respond quickly to changing workload requirements.
5. Integration with other AWS services: Auto Scaling integrates with other AWS services such as Elastic Load Balancing, Amazon CloudWatch, and AWS Identity and Access Management (IAM) to provide a comprehensive solution for scaling and managing your applications in the AWS ecosystem.

Auto Scaling is a powerful tool for ensuring the availability, performance, and cost efficiency of applications running on EC2 instances in AWS, making it a key component of many cloud-based architectures.

**QUESTIONS RELATED TO AUTO SCALING:**

**1 . If I have 3 AZ(Avalibility Zones ) 1st AZ has 12 instances 2nd has 6 instances and 3rd has 3 instances , How Autoscaling will scale this type of things?**

When using Auto Scaling in Amazon Web Services (AWS) with instances in multiple Availability Zones (AZs), the default behaviour is to distribute the instances evenly across all the AZs in the Auto Scaling group. This means that when scaling up (adding instances), Auto Scaling will launch new instances in the AZ with the fewest instances, and when scaling down (removing instances), it will terminate instances in the AZ with the most instances, in order to maintain an even distribution of instances across all the AZs.

For example, if you have 12 instances in the first AZ, 6 instances in the second AZ, and 3 instances in the third AZ, and you configure a scaling policy to scale up the Auto Scaling group, Auto Scaling will launch new instances in the third AZ to balance the number of instances across all the AZs. Similarly, if you configure a scaling policy to scale down the Auto Scaling group, Auto Scaling will terminate instances in the first AZ, which has the most instances, to maintain an even distribution of instances.

This default behaviour can be modified by using a feature called "AZ Rebalance" in Auto Scaling. When AZ Rebalance is enabled, Auto Scaling will automatically rebalance the instances across AZs during scale-in or scale-out events, regardless of the default behaviour. This helps to ensure that the instances are evenly distributed across all AZs in the Auto Scaling group, optimizing availability and fault tolerance.

It's important to note that the specific behaviour of Auto Scaling can be customized through scaling policies and configuration settings, and the actual scaling behavior will depend on the settings and policies you have defined in your Auto Scaling group.

**2 . What is AZ Rebalancing?**

AZ Rebalancing is a feature in Amazon EC2 Auto Scaling that helps to automatically distribute instances evenly across multiple Availability Zones (AZs) within an Auto Scaling group.

When you configure an Auto Scaling group with instances in multiple AZs, EC2 Auto Scaling distributes instances evenly across those AZs by default. However, over time, due to scaling events or instance failures, the distribution of instances across AZs may become uneven. Some AZs may have more instances than others, which can impact the availability and fault tolerance of your application.

**3. What are the component required to create Auto scaling?**

* Launch Configuration or Launch Template
* Auto Scaling Group
* Availability Zones (AZs)
* Scaling Policies (optional)
* Load Balancer (optional)
* CloudWatch Alarms (optional)
* Notification Actions (optional)
* Launch Configuration or Launch Template: This is the configuration that defines the specifications for the instances that will be launched by the Auto Scaling group, such as the instance type, AMI, and other settings.
* Auto Scaling Group: This is the main component that represents a collection of instances in AWS that are automatically scaled based on the defined scaling policies. It defines the minimum, maximum, and desired number of instances in the group, as well as the scaling policies and health checks.
* Availability Zones (AZs): Auto Scaling groups can span multiple AZs, which are distinct locations within an AWS Region that are engineered to be isolated from each other in terms of power, networking, and other infrastructure. Instances in the Auto Scaling group can be launched in one or more AZs for increased availability and fault tolerance.
* Scaling Policies (optional): These are rules that define how the Auto Scaling group should scale in or scale out based on certain conditions, such as CPU utilization, network traffic, or custom metrics. Scaling policies can be configured to automatically add or remove instances from the Auto Scaling group to meet the defined criteria.
* Load Balancer (optional): If you want to distribute traffic across instances in the Auto Scaling group, you can configure a load balancer, such as the Elastic Load Balancer (ELB), to work in conjunction with the Auto Scaling group. The load balancer distributes incoming traffic across instances, and the Auto Scaling group automatically scales the number of instances up or down based on the traffic load.
* CloudWatch Alarms (optional): CloudWatch is a monitoring service in AWS that can be used to set up alarms based on various metrics, such as CPU utilization, network traffic, or custom metrics. These alarms can be used to trigger scaling policies in the Auto Scaling group, automatically adding or removing instances based on the defined thresholds.
* Notification Actions (optional): You can configure actions to be taken when certain events occur in the Auto Scaling group, such as launching instances, terminating instances, or updating instances. These notification actions can be used to send notifications to email, SMS, or other endpoints to alert administrators or other stakeholders about changes in the Auto Scaling group.

**4.What are the conditions need to attach EC2 in Auto scaling group?**

To attach an EC2 instance to an auto scaling group, the instance needs to meet the following conditions:

1. The instance needs to be launched from an Amazon Machine Image (AMI) that is compatible with the operating system and application software used by the auto scaling group.
2. The instance needs to be launched in a subnet that is associated with the auto scaling group.
3. The instance needs to be launched with an IAM role that allows it to interact with the auto scaling group and other AWS services that are needed by the application.
4. The instance needs to have the necessary security group rules to allow communication with other instances in the auto scaling group and with external clients.
5. The instance needs to have the AWS Systems Manager agent installed and configured to allow the auto scaling group to manage the instance.
6. The instance needs to have the CloudWatch agent installed and configured to allow monitoring of the instance by the auto scaling group.
7. The instance needs to be launched with the correct instance type and other configuration settings that are compatible with the requirements of the auto scaling group and the application.
8. Instance is in the same AZ of same group.
9. If the Existent Ec2 is under the auto scaling group , exceed the max capacity of auto scaling group policy then request will fail
10. In summary, while an auto scaling group may not be able to launch new instances if it has reached its maximum capacity, it will still attempt to maintain the desired capacity by replacing unhealthy instances and scaling in when the load on the application decreases.

**5.What is grace Period in EC2 Auto scaling?**

In Amazon Elastic Compute Cloud (EC2) Auto Scaling, a grace period refers to the period of time after an instance is launched or terminated before any further scaling activities occur. During this period, Auto Scaling waits before it considers the newly launched or terminated instance to be fully operational or out of service.

The purpose of the grace period is to allow applications running on the instance to fully initialize and start accepting traffic or to finish processing any in-flight requests before additional scaling activities occur. This ensures that new instances are fully ready to handle incoming traffic before they are included in the Auto Scaling group.

By default, the grace period in EC2 Auto Scaling is 300 seconds (5 minutes). However, this value can be customized based on the needs of the application being scaled.

**6.What cause imbalance in EC2 in AZ?**

Imbalance in EC2 instances in AWS Auto Scaling across an Availability Zone (AZ) can occur due to the following reasons:

1. Insufficient capacity: Auto Scaling launches new instances in an AZ based on the availability of capacity. If there is insufficient capacity in an AZ, Auto Scaling may not launch new instances, leading to an imbalance in the AZ.
2. AZ outage: If an AZ experiences an outage, Auto Scaling may launch new instances in other AZs, leading to an imbalance in the affected AZ.
3. Placement groups: If instances in the Auto Scaling group are launched in a placement group, they are placed in the same AZ, which can lead to an imbalance if the AZ capacity is limited.
4. Scaling policies: If the scaling policies are not properly configured, it can lead to an imbalance in an AZ. For example, if the scaling policy is set to launch new instances based on CPU utilization, it may not be effective if the application workload is memory-intensive.

To prevent imbalance in EC2 instances in AWS Auto Scaling across an AZ, it is important to properly configure the scaling policies, monitor the instance utilization and performance, and ensure that the instances are properly configured and optimized for the workload. It is also important to ensure that there is sufficient capacity in each AZ and to avoid launching instances in a placement group if AZ balance is a concern. Additionally, using multiple AZs and configuring the Auto Scaling group to distribute instances evenly across them can help ensure availability and prevent imbalance.

**7.What is auto scaling policy/matrix type?**

Auto Scaling policies in AWS allow you to automatically scale your resources based on demand, by adding or removing EC2 instances from your Auto Scaling group. These policies can be based on various metrics such as CPU utilization, network traffic, and custom metrics.

There are three types of Auto Scaling policies:

1. Target tracking scaling policy
2. Step scaling policy
3. Scheduled scaling policy
4. **Target tracking scaling policy:** This policy adjusts the number of instances in the Auto Scaling group to maintain a target value for a specific metric, such as CPU utilization or network traffic. When using a target tracking policy, you specify a target value for the metric and AWS Auto Scaling adjusts the number of instances in the group in response to changes in the metric. For example, if you specify a target CPU utilization of 50%, AWS Auto Scaling will automatically add or remove instances to maintain an average CPU utilization of 50%. Target tracking policies are useful when you have predictable and steady traffic patterns.
5. **Step scaling policy:** This policy scales the number of instances in the Auto Scaling group based on a set of scaling adjustments, which are triggered when a specified CloudWatch alarm is triggered. The scaling adjustments are specified as a set of step adjustments, each with a specified range of metric values and a corresponding adjustment to the number of instances in the group. For example, you can create a step scaling policy that adds two instances when CPU utilization exceeds 70%, and removes two instances when CPU utilization drops below 40%. Step scaling policies are useful when you have unpredictable or bursty traffic patterns.
6. **Scheduled scaling policy:** This policy allows you to schedule specific scaling actions at certain times, which can be useful for predictable changes in traffic or resource usage. With a scheduled scaling policy, you specify the start time and end time for the scaling action, as well as the desired number of instances to add or remove during that time period. You can also choose to specify a recurring schedule, such as daily or weekly, for the scaling action. For example, you can create a scheduled scaling policy to add 10 instances to your Auto Scaling group every day at 9:00 AM, in anticipation of a daily spike in traffic. Scheduled scaling policies are useful when you have predictable and recurring traffic patterns.

In summary, the Target tracking scaling policy, Step scaling policy, and Scheduled scaling policy in AWS Auto Scaling provide you with flexible and customizable ways to automatically adjust the number of instances in your Auto Scaling group based on your application's workload and traffic patterns.

**8.Condition needed to attach Ec2 in Autoscaling group?**

To attach EC2 instances to an Auto Scaling group, you need to ensure that the EC2 instances meet the following conditions:

1. AMI: You need to have a valid Amazon Machine Image (AMI) from which to launch the EC2 instances. The AMI should contain the necessary software and configurations required by your application.
2. Instance type: You need to choose an EC2 instance type that meets the requirements of your application in terms of CPU, memory, and other resources.
3. Subnets: You need to launch the EC2 instances in one or more subnets that are associated with the Auto Scaling group. The subnets should be in different availability zones to improve availability and resiliency.
4. IAM role: The EC2 instances should have an IAM role associated with them that allows them to access AWS services and resources. The IAM role can be specified when launching the EC2 instances.
5. Security groups: The EC2 instances must be associated with one or more security groups that allow traffic to and from the instances. The security groups can be specified when launching the EC2 instances.
6. User data: If you need to configure the EC2 instances with additional settings or software during launch, you can use user data to pass a script or command to the instances. The user data can be specified when launching the EC2 instances.
7. Tags: You can add tags to the EC2 instances to help you identify and manage them. You can specify tags when launching the instances or later when updating the instance properties.
8. Instance state: The EC2 instances must be in a running state in order to be added to the Auto Scaling group. If the instance is stopped, it cannot be added to the group until it is started.
9. Same availability zone: The EC2 instances must be launched in the same availability zone as the Auto Scaling group. This is important to ensure that the instances are able to communicate with each other and with other resources in the same availability zone.

Once the EC2 instances meet these conditions, you can create an Auto Scaling group and attach the instances to it. The Auto Scaling group will use the specified AMI and launch configuration to create new instances as needed, based on the scaling policies you define.

**9.What are the Matrix type in AWS Auto Scaling?**

In AWS Auto Scaling, metrics are used to determine when to trigger scaling events, and scaling policies determine how the Auto Scaling group should adjust its capacity in response to those events.

There are many different types of metrics that can be used in AWS Auto Scaling, including but not limited to:

* CPU utilization: The percentage of CPU capacity in use by instances in the Auto Scaling group.
* Network traffic: The amount of incoming or outgoing traffic, measured in bytes per second or packets per second.
* Memory utilization: The percentage of memory capacity in use by instances in the Auto Scaling group.
* Disk I/O: The number of read and write operations per second on instance disks.
* Request counts: The number of incoming requests to an application or service, such as HTTP requests to a web server.

These metrics can be used with the different types of scaling policies available in AWS Auto Scaling, including Target Tracking Scaling, Step Scaling, Simple Scaling, and Scheduled Scaling.

**10.How to delete Autoa Scaling Group with out deleting instances (EC2)?**

To delete an AWS Auto Scaling group without deleting the instances, you can follow these steps:

1. Navigate to the AWS Management Console and open the EC2 Auto Scaling service.
2. In the left-hand navigation pane, select "Auto Scaling Groups".
3. Select the Auto Scaling group that you want to delete.
4. Click the "Actions" button and select "Detach Instances".
5. Select the instances that you want to keep and click "Detach".
6. Once the instances are detached, you can delete the Auto Scaling group by selecting it and clicking the "Actions" button and selecting "Delete Auto Scaling Group".

Detaching the instances from the Auto Scaling group will remove them from the group, but they will continue to run as standalone EC2 instances. You can then manage these instances individually using the EC2 console or other AWS services.

It's important to note that deleting an Auto Scaling group will not delete any associated EC2 instances unless you specifically choose to terminate them during the deletion process. By detaching the instances before deleting the Auto Scaling group, you can ensure that the instances continue to run independently without interruption.

**11.Feature of ELB (ELESTRIC LOAD BALANCER) IN AUTO SCALING ?**

EBL (Elastic Beanstalk Application Load Balancer) is a feature in AWS (Amazon Web Services) that allows for automatic scaling of applications in response to changes in traffic load. When using EBL with an auto scaling group in AWS, there are several features that are available:

1. Automatic scaling: EBL in combination with an auto scaling group allows for automatic scaling of application instances based on traffic load.
2. Health checks: EBL monitors the health of the application instances in the auto scaling group and automatically routes traffic to healthy instances.
3. SSL termination: EBL can terminate SSL (Secure Sockets Layer) connections, which provides an additional layer of security for your application.
4. Application routing: EBL can route traffic to specific application versions or environments, making it easier to manage multiple versions of an application.
5. Sticky sessions: EBL can enable sticky sessions, which ensure that a user's requests are always routed to the same application instance, improving user experience.
6. Monitoring and logging: EBL provide monitoring and logging capabilities, allowing you to view metrics such as request latency and error rates, and diagnose issues with your application.

**12.What are the 4 situation in which ASG send SNS EMAIL notification in AWS?**

In AWS, an Auto Scaling Group (ASG) can send Simple Notification Service (SNS) email notifications in the following four situations:

1. Launch: When a new instance is launched in the ASG, an SNS notification can be sent to alert you that a new instance has been added.
2. Terminate: When an instance is terminated in the ASG, an SNS notification can be sent to alert you that an instance has been terminated.
3. Health check failure: When an instance in the ASG fails a health check, an SNS notification can be sent to alert you of the failure.
4. Scaling: When the ASG scales up or down, an SNS notification can be sent to alert you of the scaling event. This includes both manual and automatic scaling events.

You can configure these notifications by creating an SNS topic and then configuring the ASG to send notifications to that topic.

**13. How to merge multiple single AZ Auto Scaling Groups (ASGs) into a single Multi-AZ ASG in AWS** **?**

To merge multiple single AZ Auto Scaling Groups (ASGs) into a single Multi-AZ ASG in AWS, you can follow these steps:

1. Create a new Multi-AZ Auto Scaling Group: Create a new Auto Scaling Group with Multi-AZ enabled and configure it to use the desired launch configuration and scaling policies.
2. Detach instances from the old ASGs: Detach the instances from the old single AZ ASGs that you want to add to the new Multi-AZ ASG. You can do this using the AWS Management Console or the AWS CLI.
3. Register instances with the new ASG: Register the detached instances with the new Multi-AZ ASG using the AWS Management Console or the AWS CLI.
4. Verify scaling policies: Verify that the scaling policies in the new ASG are properly configured to handle the additional instances and to maintain the desired levels of availability.
5. Test the new ASG: Test the new Multi-AZ ASG to ensure that it is functioning correctly and that all instances are properly registered and balanced across the available zones.
6. Remove the old ASGs: Once you have verified that the new Multi-AZ ASG is functioning correctly, you can remove the old single AZ ASGs.

Note: Before making any changes to your production environment, it is always recommended to thoroughly test the changes in a non-production environment and to have a rollback plan in place in case of any issues.

**14.When creating an Availability Zone (AZ) in AWS, why do we also create subnets, and how are the number of subnets related to the number of AZs?**

**This question would allow for a more specific answer that explains the relationship between AZs and subnets in AWS and how they work together to create a fault-tolerant architecture**