**AWS EC2**

AWS EC2 (Elastic Compute Cloud) is a web service provided by Amazon Web Services (AWS) that allows users to create and manage virtual servers in the cloud. EC2 provides scalable computing resources, enabling users to easily launch instances (virtual machines) with varying configurations to meet their specific needs.

**Key features and capabilities of AWS EC2 include:**

**1. Instance Types:** EC2 offers a wide range of instance types with varying combinations of CPU, memory, storage, and networking capacities. This allows users to select the most appropriate instance type for their specific workloads and applications.

**2. Operating Systems and AMIs:** EC2 supports a variety of operating systems, including Linux, Windows, and macOS. Users can choose from pre-configured Amazon Machine Images (AMIs) or create their own custom AMIs to launch instances with their desired software and configurations.

**3. Scalability and Auto Scaling:** EC2 instances can be easily scaled up or down based on demand. Auto Scaling allows users to define scaling policies that automatically adjust the number of instances based on predefined conditions, ensuring efficient resource utilization and application availability.

**4. Elastic Block Store (EBS):** EBS provides persistent block-level storage volumes that can be attached to EC2 instances. EBS volumes offer different performance characteristics and durability options, allowing users to store data independently of the instance's lifecycle.

**5. Security and Networking:** EC2 provides various security features, including security groups (firewall rules), Virtual Private Cloud (VPC) for isolated network environments, and the ability to configure network access control lists (ACLs), subnets, and routing tables.

**6. Load Balancing:** AWS Elastic Load Balancer (ELB) can be used to distribute incoming traffic across multiple EC2 instances, improving application availability, fault tolerance, and scalability.

EC2 is widely used for various use cases, including web hosting, application hosting, data processing, machine learning, and more. Its flexibility, scalability, and extensive feature set make it a core component of many cloud-based applications and infrastructures.

**EC2 INSTANCE TYPES:**

AWS EC2 (Elastic Compute Cloud) offers a broad range of instance types to cater to different computing requirements. Each instance type is optimized for specific workloads and offers varying combinations of CPU, memory, storage, and network performance. Here are some commonly used EC2 instance types:

**1. General Purpose Instances:**

* T3: Burstable performance instances with a balance of CPU, memory, and network resources.
* M5: Balanced compute instances with a good ratio of CPU to memory, suitable for a wide range of applications.
* M6g: General-purpose instances powered by AWS Graviton2 processors, providing cost-effective performance.

**2. Compute Optimized Instances:**

* C5: Instances optimized for compute-intensive workloads, featuring high-performance processors.
* C6g: Compute instances powered by AWS Graviton2 processors, providing high compute capabilities with cost efficiency.
* C6gn: Compute instances powered by AWS Graviton2 processors with enhanced network performance.

**3. Memory Optimized Instances:**

* R5: Memory-optimized instances designed for memory-intensive workloads, such as in-memory databases and real-time analytics.
* R6g: Memory-optimized instances powered by AWS Graviton2 processors, providing high memory capacity and performance.
* X1, X1e: Instances with extremely high memory capacity, suitable for in-memory databases and big data processing.

**4. Storage Optimized Instances:**

* I3: Storage-optimized instances with high-speed NVMe SSD storage, suitable for workloads requiring high I/O performance.
* D2: Dense storage instances optimized for large-scale data processing, data warehousing, and distributed file systems.
* H1: Instances designed for big data workloads, featuring high storage capacity for sequential data access.

**5. Accelerated Computing Instances:**

* P3: Instances equipped with powerful GPUs, ideal for machine learning, deep learning, and high-performance computing.
* G4: Instances with NVIDIA GPUs, optimized for graphics-intensive applications, gaming, and video transcoding.
* Inf1: Instances powered by AWS Inferentia chips, designed for cost-effective inference workloads using machine learning models.

**6. Burstable Performance Instances:**

* T2: Burstable instances suitable for applications with variable CPU usage and cost optimization.
* T3a: Burstable instances powered by AWS Graviton2 processors, offering a balance of cost and performance.
* T4g: Burstable instances powered by AWS Graviton2 processors with improved performance and cost efficiency.

**EC2 INSTANCE STATE:**

EC2 (Elastic Compute Cloud) instances in AWS can have different states depending on their current status or lifecycle stage. Here are the common EC2 instance states:

**1. Pending:** When an EC2 instance is in the "Pending" state, it means that the request to launch the instance has been received, but it is not yet running. The instance is being prepared, and its resources, such as CPU, memory, and storage, are being allocated.

**2. Running:** The "Running" state indicates that the EC2 instance is up and running. It has been successfully launched and is ready to accept and process requests.

**3. Stopped:** When an EC2 instance is in the "Stopped" state, it is no longer running, but its underlying resources, such as the instance storage and IP address, are preserved. This state allows you to stop and start instances without terminating them, which can be useful for cost savings or maintenance purposes.

**4. Terminated:** The "Terminated" state signifies that the EC2 instance has been permanently shut down and all its resources have been released. Once an instance is terminated, it cannot be started again, and you would need to launch a new instance if required.

**5. Stopping:** The "Stopping" state indicates that the EC2 instance is in the process of being stopped. It is transitioning from the running state to the stopped state. During this transition, the instance is being shut down gracefully, and any necessary cleanup tasks are performed.

**6. Starting:** The "Starting" state signifies that the EC2 instance is in the process of being started from a stopped state. It is transitioning from the stopped state to the running state. During this transition, the necessary resources are being allocated, and the instance is being prepared for running.

**7. Rebooting:** The "Rebooting" state means that the EC2 instance is being rebooted. Rebooting involves restarting the instance's operating system without changing its underlying resources or terminating the instance.

**AWS INSTANCE PURCHASING OPTIONS**

When it comes to purchasing AWS EC2 (Elastic Compute Cloud) instances, there are different pricing options available based on your requirements. Here are the common purchasing options for EC2 instances in AWS:

**1. On-Demand Instances:**

On-Demand instances allow you to pay for compute capacity on an hourly basis, with no long-term commitments or upfront costs. This option is ideal for applications with unpredictable workloads, short-term projects, or scenarios where you need flexibility to start and stop instances as needed.

**2. Reserved Instances (RI):**

Reserved Instances provide a significant cost-saving option for steady-state workloads. With RIs, you commit to a specific instance type, region, and term length (one to three years) in exchange for a discounted hourly rate compared to On-Demand instances. RIs offer different payment options: All Upfront, Partial Upfront, or No Upfront.

* Standard Reserved Instances: These offer the highest discount but have the least flexibility in terms of instance size changes within the same instance family.
* Convertible Reserved Instances: These provide a smaller discount but allow you to change instance attributes like instance type, operating system, or tenancy.

**3. Spot Instances:**

Spot Instances offer spare EC2 capacity at significantly reduced prices compared to On-Demand instances. Spot prices fluctuate based on supply and demand, and instances can be terminated if the Spot price rises above your bid. This option is suitable for workloads with flexible start and end times, and it can provide significant cost savings when used strategically.

**4. Dedicated Hosts:**

Dedicated Hosts provide physical EC2 servers fully dedicated to your use. With this option, you have complete control over the underlying host hardware and instance placement. Dedicated Hosts are suitable for compliance requirements, licensing restrictions, or for instances with high-performance needs.

**Note:**

**OnDemanded Instance**

* Whenever we want then we can create it
* Fixed Price (Hourly)
* Pay For Use.
* No Prior Payments
* No Commitment.

**Reserved Instances**

* It is like advanced booking.
* Long Term Commitment.
* Prior Payment option available ( partial payment / full payment )
* Commitment for 1 year or 3 years
* AWS will provide discount on price

**Spot Instances**

* It is like bidding or Auction.
* AWS will offer high capacity systems for low price.
* 72% savings on price.

**Dedicated Host Instance**

* This is a physical machine
* Licensed Software.
* It is very costly when compared with other instance types

**Q) What is Key Pair in EC2?**

1. Key Pair is the combination of Public Key and Private Key.
2. AWS will store public key and it will provide private key for us (We have to save that)
3. Private Key file extension will be .pem.
4. Public key & Private Key is used to connect with EC2 instance securely.
5. If we want to connect with EC2 VM we need to provide private key for AWS then AWS will compare our private key with its public key. If both keys are matched then only AWS will allow to connect with EC2.

**Note:**

* One key pair we can use to create multiple instances
* Key-Pairs are free of cost (No Bill)
* Key pair we can use for any OS Based VM.

**SECURITY GROUPS**

* Security Group is like a virtual firewall for our EC2 instance
* Security Group will control Incoming and Outgoing traffic of our EC2 instance
* To allow the incoming traffic we will configure Inbound Rules
* To allow the outgoing traffic we will configure Outbound Rules

**Example: (Service For => Protocol: Port Number)**

* **Windows => RDP: 3389**
* **Linux => SSH : 22**
* **Webserver => HTTP : 80**
* **HTTPS => HTTPS : 443**

**Note:**

* Security Groups are free to create and use.
* One security group we can use for Multiple Instances.

**CREATING WINDOWS VIRTUAL MACHINE**

1. Go to EC2 service, then click on Launch instance
2. Select AMI ( Windows - Free tier eligible)
3. Select Instance Type ( t2.micro - Free tier eligible)
4. Choose Storage (Default 30 GB - Free tier )
5. Select Network (Default VPC)
6. Create New Pair and then Create New security group with RDP protocol

**Once EC2 VM created then click on 'Connect' button and get below details**

**DNS:** ec2-15-207-89-254.ap-south-1.compute.amazonaws.com

**Username:** Administrator

**Password:** rcAF2=D3s%g&%O98o?)\*xUYd&!vdw?dp

**Connect to Windows VM using RDP**

**LAUNCHING LINUX VIRTUAL MACHINE IN AWS**

1. Go to EC2 service, then click on Launch instance
2. Select AMI ( Amazon Linux - Free tier eligible)
3. Select Instance Type ( t2.micro - Free tier eligible)
4. Choose Storage (Default 8 GB - Free tier )
5. Select Network (Default VPC)
6. Create / Use Key Pair and Create New security group with SSH protocol

Once EC2 VM created then click on 'Connect' button and get details, connect to Linux VM using MobaXterm.

**TYPES OF IP'S IN AWS**

1. IP stands for Internet Protocol
2. Every Machine should have one IP address
3. IP is like an address for computer
4. AWS providing 3 types of IPs

* Private IP
* Public IP
* Elastic IP (Static IP/Fixed Public IP)
* When we launch EC2 instance then AWS will provide one Private IP and one Public IP for our instance.
* Private IP is a fixed IP and it is used by AWS for internal purpose. It will not change when we re-start our EC2 instance.
* Public IP is a dynamic IP. When we re-start (stop and start) our EC2 instance new Public IP will be generated.

**Note:** To connect with EC2 instance from outside we will use Public IP.

* Elastic IP means fixed Public IP address.
* We can create Elastic IP and we can associate that elastic IP for our EC2 instance
* Elastic IP address will not change when we re-start our ec2 instance

**Note:** Elastic IPs are commercial (paid)

**Working process**

1) Create Elastic IP.

2) Associate Elastic IP for EC2 instance.

3) If we don't want to use Elastic IP then De-Associate Elastic IP from EC2 instance.

4) Release Elastic IP to AWS (Its mandatory).

**Note:**  If we create Elastic IP then we have to associate that Elastic IP otherwise bill will be generated for that. (We shouldn't keep Elastic IP as un-used IP)

**Load Balancing**

An AWS Load Balancer is a service provided by Amazon Web Services (AWS) that helps ***distribute incoming network traffic*** across multiple targets, such as Amazon EC2 instances, containers, or IP addresses. It helps improve the availability and scalability of applications by evenly distributing the load among the available resources.

If we deploy our application in one server then burden will increase on that server.

If burden increased on server then below are the problems we are going face.

1) Request processing will become slow.

2) Responses will be delayed for customers.

3) Server might get crash.

4) Brand / Trust issues on our business.

5) Revenue Loss.

6) Single point of failure.

* Good Business needs Good website.
* To overcome all these problems we will run our application in Multiple Servers.
* The process of running our application in Multiple Servers is called Scaling and distribute request in equal manner is called as Load Balancing.
* To implement Load Balancing we will use Load Balancer (ELB) in AWS
* LBR will receive the request and it will distribute the requests to servers in *round robin fashion*

**Types of Load Balancers in AWS**

1) Application Load Balancer (ALB)

2) Network Load Balancer (NLB)

3) Gateway Load Balancer (GLB)

To implement load balancing for HTTP & HTTPS requests we will go for Application Load Balancer (ALB)

By using Application Load Balancer we can implement Path Based Routing

**IMPLEMENTING LOAD BALANCER**

**1) Create 1st EC2 instance with below user data script**

#! /bin/bash

sudo su

yum install httpd -y

cd /var/www/html

echo "<html><h1>Welcome to AWS CLOUD :: Server 1</h1></html>" > index.html

service httpd start

**2) Create 2nd EC2 instance with below user data script**

#! /bin/bash

sudo su

yum install httpd -y

cd /var/www/html

echo "<html><h1>Welcome to AWS CLOUD :: Server 2</h1></html>" > index.html

service httpd start

**3) Create 3rd EC2 instance with below user data script**

#! /bin/bash

sudo su

yum install httpd -y

cd /var/www/html

echo "<html><h1>Welcome to AWS CLOUD :: Server 3</h1></html>" > index.html

service httpd start

4) Create Target Group with above 3 EC2 instances

(Target Group means group of servers which are running our application)

5) Create Application Load Balancer using Target Group

6) Access the application Load Balancer DNS URL

Note: When request comes to Load Balancer it will distribute the requests to servers which are part of given target group.

**Auto Scaling Group [ASG]**

ASG stands for Auto Scaling Group, and it is a feature provided by Amazon Web Services (AWS) for automatically adjusting the number of instances in a group to match the desired capacity, based on defined conditions or metrics. Auto Scaling Groups are commonly used in AWS to ensure the availability and scalability of applications running on EC2 instances.

**Points about Auto Scaling Groups (ASGs),**

**1. Scaling Policies:** ASGs use scaling policies to determine when and how to scale instances. Scaling policies define the conditions that trigger scaling actions, such as increasing or decreasing the number of instances.

**2. Desired Capacity:** ASGs have a desired capacity parameter, which represents the number of instances that should be running at any given time. The ASG automatically adjusts the number of instances to match the desired capacity.

**3. Launch Templates:** ASGs are associated with launch configurations or launch templates, which define the instance type, AMI (Amazon Machine Image), and other instance launch parameters. These configurations/templates are used to launch new instances as needed.

**4. Availability Zones:** ASGs distribute instances across multiple Availability Zones (AZs) within a region to improve application availability and fault tolerance. If an AZ becomes unavailable, the ASG automatically launches instances in a healthy AZ.

**5. Elastic Load Balancers:** ASGs can be associated with Elastic Load Balancers (ELBs) or Application Load Balancers (ALBs) to evenly distribute traffic across instances. This helps in load balancing and achieving high availability.

* AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.
* Using AWS Auto Scaling, it is easy to setup application scaling for multiple resources across multiple services in minutes.
* Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application.

**Advantages with Auto Scaling**

1) Fault Tolerance

2) Availability

3) Cost Management

**How to setup Auto Scaling Group**

Step 1) Create Launch Template

Step 2) Create Auto Scaling Group with Launch Template

Step 3) Configure Desired, Min and Max Capacity

Step 4) Attach Auto Scaling Group to particular Target Group

Step 5) Configure Scaling Policy.