UNIT 1

1. Implement Identity and Access Management (IAM) by explaining the process of managing users and groups, including the necessary steps.

Implementing Identity and Access Management (IAM): Managing Users and Groups

AWS Identity and Access Management (IAM) is a service that helps you securely manage access to AWS resources. It allows you to create users, organize them into groups, and assign permissions using policies.

Steps to Manage Users and Groups in IAM

---

1. Creating IAM Users

IAM users represent individuals or applications that need access to AWS resources. Each user has its own credentials and permissions.

Steps to Create an IAM User:

1. Go to the IAM Console in AWS.
2. In the left navigation pane, select Users → Click on Add user.
3. Enter a User name and select the type of access:
    - Programmatic Access: For API, CLI, or SDK access.
    - AWS Management Console Access: For access to the AWS Management Console.
4. Click Next to assign permissions.
5. Choose how to attach permissions:
    - Attach Policies Directly: Select pre-existing AWS policies.
    - Add to a Group: Attach the user to an existing group.
    - Copy Permissions: Copy permissions from an existing user.
6. Review the user details and click Create user.
7. Save the user credentials, including the Access Key ID and Secret Access Key, if programmatic access was granted.

---

2. Creating IAM Groups

Groups allow you to manage permissions for multiple users by assigning policies to the group instead of individual users.

Steps to Create an IAM Group:

1. Go to the IAM Console → Select Groups in the left panel.
2. Click on Create New Group.
3. Enter a Group name.
4. Attach a policy to the group:
   - Select a managed policy, such as `AmazonS3FullAccess` or `AdministratorAccess`.
   - Alternatively, create and attach a custom policy.
5. Review the group details and click Create Group.

---

3. Adding Users to Groups

To simplify permission management, users can be added to groups.

Steps to Add Users to Groups:

1. Go to the IAM Console → Select Groups.
2. Select the group to which you want to add users.
3. Click on Add Users to Group.
4. Select the IAM users you want to add and click Add users.

By adding users to a group, all users in that group inherit the permissions attached to the group.

---

4. Attaching Policies to Users or Groups

Permissions are granted to users or groups using IAM policies. Policies are written in JSON format and define what actions are allowed or denied for specific AWS resources.

Steps to Attach Policies:

1. Go to the IAM Console → Select Users or Groups.
2. Click on the user or group name.
3. Under the Permissions tab, click Add Permissions.
4. Choose one of the following:
   - Attach existing policies: Select AWS managed or custom policies.
   - Create a policy: Write a custom JSON policy.

Example Policy: Granting S3 Read-Only Access

json

Copy code

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::example-bucket/*"
    }
  ]
}
```

---

5. Configuring Multi-Factor Authentication (MFA)

For added security, enable Multi-Factor Authentication (MFA) for IAM users. MFA requires users to provide an authentication code in addition to their password when signing in.

Steps to Enable MFA for a User:

1. Go to the IAM Console → Select Users.
2. Click on the username → Go to the Security credentials tab.
3. Under Assigned MFA Device, click Manage.
4. Choose the type of MFA device:
    - Virtual MFA Device: Use a mobile app like Google Authenticator or Authy.
    - Hardware MFA Device: Use a physical MFA device.
5. Follow the instructions to activate the device and save the configuration.

---

6. Managing IAM Roles

IAM roles allow you to grant permissions to AWS services, applications, or users in other accounts without creating IAM users.

Steps to Create an IAM Role:

1. Go to the IAM Console → Select Roles → Click Create Role.
2. Choose the trusted entity:
    - AWS service (e.g., EC2)
    - Another AWS account
    - Web identity (e.g., Cognito)
3. Attach policies to define permissions.
4. Review and create the role.

Example Use Case: Assign an IAM role to an EC2 instance to allow it to access S3 without hardcoding credentials.

---

7. Setting IAM Password Policies

IAM allows you to enforce strong password policies for users.

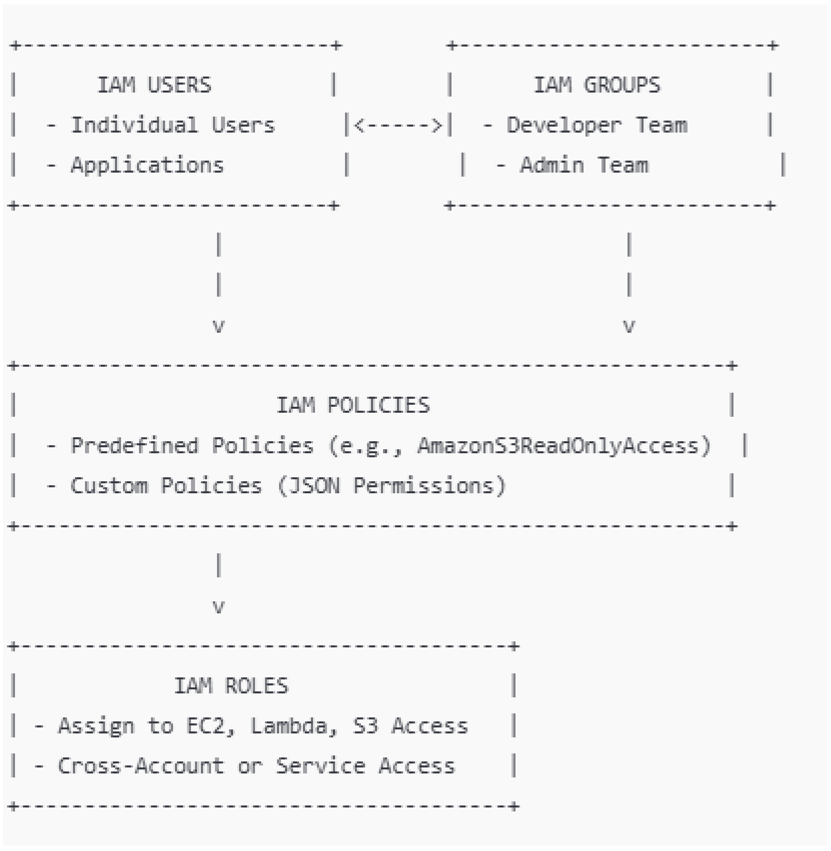
Steps to Configure Password Policy:

1. Go to the IAM Console → Select Account Settings.
2. Under Password Policy, configure the following:
    - Minimum password length.
    - Require uppercase, lowercase, numbers, and symbols.

    - Password expiration and reuse prevention.
3. Save the password policy.

Summary of IAM Workflow

1. Create IAM users for individuals or applications.
2. Create groups and assign policies to manage permissions for multiple users.
3. Attach policies (managed or custom) to users or groups to grant access.
4. Enable MFA to improve account security.
5. Create IAM roles for temporary or cross-account access.
6. Enforce password policies for better security.



```
+-----------------------+       +-----------------------+
|      IAM USERS        |       |      IAM GROUPS       |
|  - Individual Users    |<----->|  - Developer Team      |
|  - Applications        |        |  - Admin Team          |
+-----------------------+       +-----------------------+
              |                              |
              |                              |
              v                              v
+-----------------------------------------------------+
|                  IAM POLICIES                       |
|  - Predefined Policies (e.g., AmazonS3ReadOnlyAccess)  |
|  - Custom Policies (JSON Permissions)               |
+-----------------------------------------------------+
              |
              v
+------------------------------------+
|           IAM ROLES                |
| - Assign to EC2, Lambda, S3 Access  |
| - Cross-Account or Service Access   |
+------------------------------------+
```

Conclusion

AWS IAM provides a comprehensive solution for managing access to AWS resources. By creating users, groups, and roles, and attaching appropriate policies, organizations can control who has access to their AWS environment and what actions they are permitted to perform.

UNIT-2

1.Demonstrate the process of creating a static website in S3, including details on S3 storage classes and the S3 replication policy.

Creating a Static Website in Amazon S3

Amazon S3 (Simple Storage Service) is a scalable storage service that can also be used to host static websites. This involves uploading files (HTML, CSS, JavaScript, images) to an S3 bucket, configuring the bucket for static website hosting, and ensuring the correct permissions and settings.

Step-by-Step Process of Creating a Static Website in S3

1. Create an S3 Bucket

1. Log in to AWS Console:
   - Navigate to S3 from the AWS Management Console.
2. Create a New Bucket:
   - Click on Create bucket.
   - Enter a unique name for the bucket (e.g., `my-static-website-bucket`).
   - Select the appropriate AWS Region where you want to store the bucket.
   - Leave the other default settings as is or adjust them as needed.
   - Click Create.

2. Enable Static Website Hosting

1. Select the S3 Bucket:

    - In the S3 dashboard, click on the bucket you just created.
2. Enable Static Website Hosting:
    - Go to the Properties tab.
    - Scroll down to the Static website hosting section.
    - Click Edit.
    - Choose Enable.
    - Enter the Index document name (usually `index.html`).
    - Enter the Error document name (usually `error.html`).
    - Click Save changes.

3. Upload Website Files

1. Upload Your Website Files:
    - In the Objects tab of your bucket, click Upload.
    - Add your website files (e.g., HTML, CSS, images).
    - Click Upload to complete the process.
2. Set Permissions for Files:
    - You need to make your files publicly accessible. You can do this by setting public-read permissions.
    - When uploading files, in the Permissions section, ensure you select Grant public read access to this object.
3. Alternatively, for the entire bucket:
    - Go to Permissions → Bucket Policy.

Add a policy that grants public read access:
json
Copy code
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource":
"arn:aws:s3:::my-static-website-bucket/*"
```

```
    }
  ]
}
```

-
- Replace `my-static-website-bucket` with the actual name of your bucket.

4. Access the Static Website

1. Find the Website Endpoint:
   - After enabling static website hosting, the Endpoint URL will be displayed under the Static website hosting section.
   - This URL is where your static website can be accessed (e.g., `http://my-static-website-bucket.s3-website-us-west-2.amazonaws.com`).

S3 Storage Classes for Hosting

Amazon S3 offers different storage classes that can be used based on the frequency of access to the files and cost considerations.

1. S3 Standard:
   - Default and most commonly used storage class.
   - Suitable for frequently accessed data.
   - Low-latency and high-throughput performance.
2. S3 Intelligent-Tiering:
   - Automatically moves data between two access tiers (frequent and infrequent) based on usage patterns.
   - Optimized for unpredictable access patterns.
3. S3 Standard-IA (Infrequent Access):
   - Lower cost for data that is accessed less frequently but still requires rapid access.
   - Best for data that is not frequently accessed but needs to be available immediately when required.
4. S3 One Zone-IA:

    - Lower cost than Standard-IA, but data is stored in only one Availability Zone.
    - Suitable for data that can be recreated if lost.
5. S3 Glacier & S3 Glacier Deep Archive:
    - For long-term archive storage at a lower cost.
    - Retrieval times for Glacier are typically minutes to hours, and Glacier Deep Archive can take up to 12 hours.

Note: For static website hosting, S3 Standard is commonly used due to its high availability and quick access.

S3 Replication Policy

S3 offers Cross-Region Replication (CRR) and Same-Region Replication (SRR) to replicate objects across different buckets and regions. This is useful for disaster recovery, compliance, and to reduce latency for global users.

Steps to Configure S3 Replication:

1. Go to the S3 Bucket:
    - In the S3 Console, select the bucket that you want to replicate.
2. Enable Versioning:
    - Replication requires versioning to be enabled on the source and destination buckets.
    - Go to the Properties tab → Bucket Versioning → Enable.
3. Configure Replication:
    - Go to the Management tab of the bucket.
    - Under Replication, click Add rule.
    - Choose the source and destination buckets (both must have versioning enabled).
    - Set Replicate all objects or use filters to replicate specific objects.
    - Optionally, enable replicating delete markers and replicate existing objects.
    - Click Save.
4. Review and Test Replication:

    - After replication is enabled, test the replication by uploading a new file to the source bucket and checking if it appears in the destination bucket.

Replication Benefits:

- Data Redundancy: Increases availability by replicating your static website's data in multiple regions or buckets.
- Disaster Recovery: Provides a backup in case the original region or bucket experiences an issue.
- Lower Latency: Improves access speed for users in different regions by replicating content closer to them.

Diagram of S3 Static Website Hosting and Replication

```
+-------------------------+       +-------------------------+
|      IAM USERS          |       |      IAM GROUPS         |
|  - Individual Users     |<----->|  - Developer Team       |
|  - Applications         |       |  - Admin Team           |
+-------------------------+       +-------------------------+
             |                               |
             |                               |
             v                               v
+-------------------------------------------------------+
|                    IAM POLICIES                       |
|  - Predefined Policies (e.g., AmazonS3ReadOnlyAccess) |
|  - Custom Policies (JSON Permissions)                 |
+-------------------------------------------------------+
             |
             v
+--------------------------------------+
|           IAM ROLES                  |
| - Assign to EC2, Lambda, S3 Access   |
| - Cross-Account or Service Access    |
+--------------------------------------+
```

Conclusion

- Static Website Hosting in Amazon S3 is a simple and cost-effective solution for serving HTML, CSS, and JavaScript files.
- S3 Storage Classes offer flexibility in cost and performance, allowing you to choose the right option based on access frequency.
- S3 Replication ensures data redundancy, disaster recovery, and low-latency access by replicating your files across regions or buckets.

By following the steps above, you can easily create a static website in S3, apply the correct storage class, and configure S3 replication for added reliability.

UNIT-3

1.Demonstrate the process of taking a screenshot, creating an AMI, and launching an instance from that AMI in another region, utilizing EBS.

Process of Taking a Screenshot, Creating an AMI, and Launching an Instance from that AMI in Another Region in AWS Using EBS

In Amazon Web Services (AWS), you can capture the state of your EC2 instance by creating an Amazon Machine Image (AMI). This AMI includes all the configuration, installed applications, and data on the instance, allowing you to launch identical instances in any region. Below is a detailed guide on how to take a screenshot, create an AMI, and launch an instance from that AMI in another region using Elastic Block Store (EBS).

Step 1: Take a Screenshot of an EC2 Instance

In AWS, you cannot directly take a screenshot like you would on a local computer. However, if you want to capture the graphical output of an instance (for example, a desktop), you would need to install a remote desktop protocol (RDP) or Virtual Network Computing (VNC) server on the EC2 instance and connect to it.

However, if you're referring to taking a snapshot (which is a point-in-time backup of your EBS volume) rather than a screenshot, you can do the following:

Take a Snapshot of the EBS Volume

1. Log in to AWS Console: Go to the EC2 Dashboard.
2. Go to Volumes: In the left-hand navigation pane, select Volumes under Elastic Block Store.
3. Select the Volume: Find the EBS volume attached to your EC2 instance and select it.
4. Create Snapshot:
   - Click on Actions and select Create Snapshot.
   - Provide a name and description for the snapshot.
   - Click Create Snapshot.

Step 2: Create an AMI (Amazon Machine Image)

An AMI is essentially a backup of your entire EC2 instance (including its configuration, EBS volumes, and data).

Create an AMI from an EC2 Instance

1. Log in to AWS Console: Go to the EC2 Dashboard.
2. Select Instances: From the left sidebar, select Instances and choose the instance for which you want to create an AMI.
3. Create Image:
   - Right-click on the instance or use the Actions button.
   - Choose Image and Templates → Create Image.
   - Give your image a name (e.g., `My-EC2-AMI`).
   - Optionally, you can specify additional options, such as whether to include additional volumes in the AMI.
   - Click Create Image.
4. Check AMI Status: Go to the AMIs section in the Images section of EC2. The AMI will be created, and its status will be listed as pending. Once the creation process is complete, the status will be available.

Step 3: Copy AMI to Another Region

Now that you have an AMI, you can copy it to another region where you want to launch a new EC2 instance.

1. Go to AMIs Section: In the EC2 dashboard, go to AMIs under Images.

2. Select the AMI: Find and select the AMI you created earlier.
3. Copy AMI:
    - With the AMI selected, click on Actions and select Copy AMI.
    - Choose the Destination Region where you want the AMI to be copied.
    - Optionally, you can modify the AMI's name, description, or encryption settings.
    - Click Copy AMI.
4. Note: It may take several minutes to complete the AMI copy process depending on the size of the image.

Step 4: Launch an EC2 Instance from the AMI in Another Region

Once the AMI is copied to the destination region, you can launch a new EC2 instance using that AMI.

Launch Instance from AMI

1. Switch to the Target Region: In the AWS Management Console, switch to the destination region where you copied the AMI.
2. Go to EC2 Dashboard: Navigate to the EC2 Dashboard in the target region.
3. Select AMIs: Under Images, select AMIs and locate the copied AMI.
4. Launch Instance:
    - Select the AMI and click on Launch.
    - Choose the instance type based on your requirements.
    - Configure the instance (network settings, IAM role, etc.).
    - Add storage: By default, the root volume is created based on the settings in the AMI, but you can also add additional volumes.
    - Configure security groups: Choose or create a security group for your instance.
    - Review the settings and click Launch.
5. Access the Instance: Once the instance is launched, you can access it using SSH (Linux instances) or RDP (Windows instances), depending on your configuration.

Step 5: EBS Volume and Storage

- Elastic Block Store (EBS) is the storage that persists even when the instance is stopped or terminated. The root EBS volume is automatically created when you launch an EC2 instance.
- If you're copying the instance to another region, the EBS volume and data are part of the AMI, and any new instance launched from that AMI will inherit the same EBS volumes.
- Add Additional EBS Volumes: While launching a new instance, you can attach additional EBS volumes if required. This is done during the Configure Instance Details section while launching the instance, where you can click on Add Storage to attach extra volumes.

Diagram for Launching EC2 Instance with EBS in Another Region

plaintext
Copy code

```
+------------------------------------+
|       AWS Region A                 |
|                                     |
|    +-------------------------+     |
|    | EC2 Instance (Source)   |     |
|    |  + EBS Volume (Storage) |     |
|    +-------------------------+     |
|                |                   |
|    [Create AMI]                     |
|                |                   |
|    v                                |
|    +-------------------------+     |
|    | Amazon Machine Image     |     |
|    +-------------------------+     |
|                                     |
|    [Copy AMI to Region B]          |
+------------------------------------+

+------------------------------------+
|       AWS Region B                 |
|                                     |
|    +-------------------------+     |
|    | EC2 Instance (Target)   |     |
|    |  + EBS Volume (Storage) |    |
|    +-------------------------+    |
|                                   |
|    [Launch EC2 Instance]          |
+-----------------------------------+
```

Conclusion

The process of creating and using an AMI across AWS regions is a powerful way to replicate your EC2 instance configurations in different locations, enabling disaster recovery, scaling, and migration. By utilizing EBS volumes, your data remains intact during instance creation and migration, allowing for seamless data transfer between regions.

UNIT-4

1.Explain a scenario for Auto Scaling and outline the planning process, along with a clear diagram.

Scenario for AWS Auto Scaling

Scenario: E-Commerce Website Handling Seasonal Traffic

Imagine you run an e-commerce website that experiences a surge in traffic during peak seasons like Black Friday or holiday sales. During these times, the website must scale up to handle the increased demand without performance degradation. Once the traffic returns to normal, the resources should scale down to save costs.

To address this challenge, AWS Auto Scaling can be used to automatically adjust the number of EC2 instances in your application based on traffic or system performance metrics.

Auto Scaling Planning Process

1. Define the Requirements

- Identify Scaling Triggers: Determine the metrics that indicate a need to scale up or down. Common triggers include:
  - CPU utilization (e.g., above 70% for scaling out).
  - Network traffic (e.g., sudden increases in HTTP requests).
  - Application response time.
- Minimum and Maximum Capacity: Decide the minimum and maximum number of instances required for your application.

2. Create a Launch Template or Launch Configuration

- Set up the Launch Template to define:
  - EC2 instance type (e.g., t2.medium, m5.large).
  - AMI (Amazon Machine Image) to launch instances.
  - Storage configuration (e.g., EBS volumes).
  - Security Groups and IAM roles for the instances.

3. Configure Auto Scaling Group (ASG)

- Define the Auto Scaling Group to:
  - Manage the group of EC2 instances.
  - Monitor health and replace unhealthy instances.
  - Ensure the specified minimum and maximum number of instances are maintained.

4. Set Up Scaling Policies

Choose appropriate scaling policies:

- Target Tracking Policy: Automatically scales to keep a specified metric at a target value (e.g., 60% CPU utilization).
- Step Scaling Policy: Adds or removes instances based on specific thresholds (e.g., add 2 instances if CPU > 80%).
- Scheduled Scaling: Scale at specific times (e.g., Black Friday at 9 AM).

5. Monitor and Test

- Monitor the Auto Scaling behavior using Amazon CloudWatch.
- Test the setup to ensure instances scale up/down based on traffic or resource demand.

Auto Scaling Workflow

1. CloudWatch Monitors Metrics: CloudWatch continuously monitors key performance metrics like CPU utilization.
2. Trigger Scaling Policies: If a predefined threshold is breached (e.g., CPU > 70%), CloudWatch triggers the Auto Scaling policy.
3. Launch or Terminate Instances: The Auto Scaling Group automatically launches new instances or terminates instances based on demand.
4. Elastic Load Balancer (ELB): New instances are registered with the load balancer, ensuring even traffic distribution.
5. Scaling Down: When the load decreases, the Auto Scaling Group terminates excess instances to optimize costs.

Diagram for Auto Scaling Process

plaintext
Copy code

```
+-----------------------------------------------+
|                  Users/Traffic                |
+-----------------------+-----------------------+
                        |
                        v
+--------------------------------+
|    Elastic Load Balancer       |
+--------------------------------+
             |
+--------------------------------+
|    Auto Scaling Group (ASG)    |
|   - Min Instances: 2           |
|   - Max Instances: 10          |
|   - Desired Capacity: 4        |
+--------------------------------+
             |
     +---------------------+
     |    EC2 Instances    |
     +---------------------+
             |
      Monitored by CloudWatch
```

```
                |
   Scaling Policies Triggered
```

Conclusion

In this scenario:

- AWS Auto Scaling helps the e-commerce website handle fluctuating traffic efficiently.
- Resources scale up automatically during peak loads and scale down when traffic decreases, optimizing performance and cost.
- Monitoring tools like CloudWatch ensure that scaling actions are triggered based on predefined performance metrics.

UNIT-5
1. Discuss in detail the working procedure of DNS in application, along with a clear diagram.

Working Procedure of DNS in Applications

The Domain Name System (DNS) is a hierarchical and decentralized naming system that translates human-readable domain names (like `www.example.com`) into IP addresses (like `192.0.2.1`) that computers use to identify each other on the internet.

How DNS Works in Applications

When you type a domain name (e.g., `www.example.com`) into a web browser or an application tries to connect to a domain, the DNS resolution process begins. Here's the step-by-step working procedure:

Step-by-Step DNS Resolution Process

1. User Request:
   - A user enters a domain name (`www.example.com`) in a web browser or application.
   - The system first checks its local cache (e.g., browser or operating system cache) to see if the IP address for the domain is already stored.
2. Query the Recursive Resolver:
   - If the IP address is not in the cache, the query is forwarded to a DNS Recursive Resolver. This is typically provided by the user's Internet Service Provider (ISP) or a public DNS service like Google DNS (8.8.8.8).
3. Query the Root Name Server:
   - The Recursive Resolver sends the query to a Root Name Server, which knows the location of Top-Level Domain (TLD) servers.
   - The Root Name Server responds with the address of the TLD Server (e.g., `.com` servers for `example.com`).
4. Query the TLD Server:
   - The Recursive Resolver sends the query to the TLD Server (e.g., `.com` server).
   - The TLD Server responds with the address of the Authoritative Name Server for the domain (`example.com`).
5. Query the Authoritative Name Server:
   - The Recursive Resolver queries the Authoritative Name Server, which holds the actual IP address of `www.example.com`.
   - The Authoritative Name Server responds with the IP address (e.g., `192.0.2.1`).
6. Return the IP Address:
   - The Recursive Resolver returns the IP address to the client (browser or application).
7. Connect to the Web Server:
   - The browser or application uses the IP address to connect to the target server hosting the website or application.
   - The requested content (e.g., a web page) is delivered to the user.

Diagram of DNS Resolution Process

plaintext

Copy code

```
+--------------------------------+
|         User/Browser           |
|        (www.example.com)       |
+--------------------------------+
                |
                v
+--------------------------------+
|      DNS Recursive Resolver    |
|      (ISP or Public DNS)       |
+--------------------------------+
                |
      Query Root Name Server
                |
                v
+--------------------------------+
|       Root Name Server         |
|       (Points to TLD)          |
+--------------------------------+
                |
      Query TLD Server (.com)
                |
                v
+--------------------------------+
|         TLD Name Server          |
|    (Points to Authoritative)     |
+--------------------------------+
                |
Query Authoritative Name Server
                |
                v
+--------------------------------+
|    Authoritative Name Server    |
|    (Provides IP Address)        |
+--------------------------------+
                |
        Return IP Address
                |
                v
+--------------------------------+
|       User/Browser Connects     |
|     (www.example.com --> IP)    |
+--------------------------------+
                |
        Content is Delivered
```

Key Components in DNS Resolution

1. DNS Recursive Resolver:

    - Acts as a middleman between the user and DNS servers.
    - Performs the DNS lookup process.
2. Root Name Server:
    - Top of the DNS hierarchy.
    - Directs queries to the appropriate TLD servers.
3. TLD Name Server:
    - Handles requests for domain extensions like `.com`, `.org`, `.net`, etc.
4. Authoritative Name Server:
    - Stores the actual IP address for the domain name.
5. Browser Cache/OS Cache:
    - Stores previously resolved DNS queries for faster lookup.

DNS in Applications

- Web Browsing: Browsers use DNS to resolve domain names to IP addresses before connecting to web servers.
- Email Delivery: Applications like email clients use DNS to look up Mail Exchange (MX) records to find mail servers.
- APIs and Microservices: Applications call other services using domain names, which are resolved to service endpoints via DNS.
- Content Delivery Networks (CDN): DNS resolves the domain to the nearest CDN edge location for fast content delivery.

Conclusion

DNS plays a critical role in allowing users and applications to interact with servers using human-readable domain names. By resolving these names into IP addresses through the recursive query process, DNS ensures seamless and efficient communication across the internet.