Creating and managing cloud resources on AWS a) tour of aws

Tour of AWS

1. AWS Management Console
    - Web-based interface for managing AWS services.
    - Provides a unified dashboard for monitoring and configuration.
2. AWS Global Infrastructure
    - Regions: Physical locations with multiple Availability Zones.
    - Availability Zones (AZs): Data centers within a region for high availability.
    - Edge Locations: Used for content delivery via AWS CloudFront.
3. Core AWS Services
    - Compute: EC2, Lambda, ECS, EKS
    - Storage: S3, EBS, Glacier
    - Networking: VPC, Route 53, CloudFront
    - Databases: RDS, DynamoDB, Aurora
    - Security & IAM: IAM, KMS, WAF
4. Billing & Pricing
    - Pay-as-you-go model
    - Free Tier options available
    - Cost Explorer for expense tracking
5. Monitoring & Management
    - AWS CloudWatch: Logs and metrics monitoring
    - AWS CloudTrail: Tracks API calls for security auditing
    - AWS Config: Ensures compliance
6. Deployment & Automation
    - AWS CloudFormation: Infrastructure as Code (IaC)
    - AWS Elastic Beanstalk: Automated deployment
    - AWS Lambda: Serverless computing
7. Security Best Practices
    - Use IAM roles and policies
    - Enable MFA for accounts
    - Encrypt data in transit and at rest

IAM

AWS Identity and Access Management (IAM) – Steps

1. Access AWS IAM
    - Sign in to AWS Management Console.
    - Navigate to IAM service.
2. Create a New User
    - Click Users → Add user.
    - Enter Username (e.g., test-user).
    - Select Access type:
        - Programmatic Access (for CLI/SDK/API).
        - AWS Management Console Access (set a custom password).
    - Click Next: Permissions.
3. Assign Permissions
    - Add user to Group (assign policies like AdministratorAccess, AmazonS3FullAccess).
    - OR attach policies directly to the user.
    - Click Next: Tags.
4. Add Tags (Optional)
    - Add key-value pairs for organization.
    - Click Next: Review.
5. Review and Create User
    - Verify details, click Create user.
    - Download .csv file (contains Access Key ID & Secret Access Key).
6. Create and Attach a Role
    - Go to Roles → Click Create role.
    - Select AWS Service (e.g., EC2).
    - Attach necessary policies (e.g., AmazonS3FullAccess).
    - Name the role and create it.
    - Attach the role to an AWS service (e.g., EC2 instance).
7. Review IAM Policies and Groups
    - Regularly review permissions to ensure least privilege access.

INTRODUCTION TO AMAZON EC2

Getting Started with Amazon EC2 (Elastic Compute Cloud)

Amazon EC2 is a web service that provides resizable compute capacity in the cloud, enabling developers to scale their applications easily.

Steps to Get Started with Amazon EC2:

1. Access the AWS Management Console

- Sign in to the AWS Management Console.
- Navigate to EC2 under the "Compute" section.

2. Launch an EC2 Instance

- Click on **Launch Instance**.
- **Choose an Amazon Machine Image (AMI):**
    - Select an AMI from the list (includes OS and pre-installed applications).
    - Popular choices: **Amazon Linux 2, Ubuntu, Windows Server**.
- **Choose an Instance Type:**
    - Select an instance type based on your needs.
    - Common types: **t2.micro (free-tier eligible), m5.large**, etc.
    - Click **Next: Configure Instance Details**.
- **Configure Instance Details:**
    - Set the number of instances, network settings, IAM roles, and other advanced options.
    - Click **Next: Add Storage**.
- **Add Storage:**
    - Define the size and type of storage (EBS volumes) to attach.
    - Default settings are sufficient for basic setups.
    - Click **Next: Add Tags**.
- **Add Tags:**
    - Optionally, add tags (key-value pairs) to help manage and identify your instance.
    - Click **Next: Configure Security Group**.
- **Configure Security Group:**
    - A security group acts as a virtual firewall.
    - Set rules to control traffic to your instance.
    - For basic setups, add an SSH rule to allow access from your IP.
    - Click **Review and Launch**.
- **Review and Launch:**
    - Review instance settings and click **Launch**.
    - Create or select an SSH key pair for secure access.
    - Choose **Create a new key pair** or **Use an existing one**, then click **Launch Instances**.

3. Connect to Your EC2 Instance

- Once the instance is running, click **View Instances** to see details.

INTRODUC TO AMAZON SIMPLE STORAGE SERVICE S3

Amazon S3: Getting Started Guide
1. Access the AWS Management Console
- Sign in to the AWS Management Console.
- Navigate to S3 under the "Storage" section.

2. Create an S3 Bucket
- Click on Create bucket.
- Bucket Name: Enter a globally unique name for your bucket.
- Region: Choose an AWS Region where the bucket will reside.
- Bucket Settings: Configure settings like versioning, server access logging, and encryption (these can be adjusted later).
- Block Public Access: By default, S3 buckets block public access. Adjust if needed.
- Click Create Bucket.

3. Upload Objects to S3
- Click on your newly created bucket.
- Click Upload.
- Drag and drop files or use the file picker to select files from your local machine.
- Set object properties like storage class and encryption if needed.
- Click Upload to store the files in S3.

4. Manage Bucket Permissions
- Go to the Permissions tab in your bucket.
- Manage access using:
  - Bucket policies
  - Access control lists (ACLs)
  - IAM roles
- To make files public, modify the bucket policy or set specific permissions on individual objects.

5. Enable Versioning
- Versioning helps keep multiple versions of an object in one bucket. To enable:
  - Go to Bucket Settings → Properties → Versioning.
  - Click Edit and enable versioning.

6. Set Up Lifecycle Policies
- Lifecycle policies allow automatic transitioning of objects between storage classes or deletion after a period.
- Go to the Management tab and click Create lifecycle rule.
- Define rules to transition objects to cheaper storage classes (e.g., Glacier) or delete them after a set time.

7. Access Objects in S3
- Objects can be accessed via the AWS S3 Console or programmatically using the AWS SDKs or CLI.
- To generate a pre-signed URL for temporary access:
- aws s3 presign s3://bucket-name/object-key --expires-in 3600

8. Delete S3 Bucket
- To delete a bucket, first delete all objects within it.
- Then, delete the bucket itself through the AWS Console or CLI.

CREATING AND RUNNING CONTAINERS

A Container That Can Run Consistently Across Different Environments
Steps to Create and Run Containers:

1. Install Docker:
   - Install Docker by following the instructions on the Docker website.
   - Verify the installation:
   - docker --version
2. Create a Dockerfile:
   - A Dockerfile is a text document that contains all the commands to assemble a container image.
   - Example Dockerfile for a simple Node.js application:
   - # Use an official Node.js runtime as a parent image
   - FROM node:14
   -
   - # Set the working directory in the container
   - WORKDIR /usr/src/app
   -
   - # Copy the local files to the container
   - COPY package*.json ./
   -
   - # Install any needed dependencies
   - RUN npm install
   -
   - # Copy the rest of the application code
   - COPY . .
   -
   - # Make port 8080 available to the world outside this container
   - EXPOSE 8080
   -
   - # Run the app when the container launches
   - CMD ["node", "app.js"]
3. Build the Docker Image:
   - Navigate to the directory containing the Dockerfile and run:
   - docker build -t my-node-app .
   - This command builds the Docker image and tags it as my-node-app.
4. Run the Docker Container:
   - Use the following command to run the container:
   - docker run -p 8080:8080 my-node-app
   - The -p 8080:8080 flag maps port 8080 on your local machine to port 8080 in the container.
5. Manage Containers:
   - To list running containers:
   - docker ps
   - To stop a container:
   - docker stop <container_id>
   - To remove a container:
   - docker rm <container_id>
   - To remove an image:
   - docker rmi <image_id>
6. Using Docker Compose (Optional):
   - For complex applications with multiple services, use

- **To connect via SSH:**
  - Select the instance, click **Connect**.
  - Follow the provided instructions, typically using:
  - ssh -i /path/to/key-pair.pem ec2-user@your-ec2-public-dns
- **For Windows instances**, connect using RDP (Remote Desktop Protocol).

4. Manage and Monitor the Instance
- Use **Instance State** actions to **start, stop, reboot, or terminate** the instance.
- Monitor performance via **CloudWatch** (CPU usage, network traffic, etc.).

5. Terminate the Instance
- To avoid charges, **terminate** the instance when it is no longer needed.

AMAZON REALTIONAL DATABASE SERVICE

Amazon RDS (Relational Database Service) makes it easy to set up, operate, and scale a relational database in the cloud. With RDS, you can run several database engines, including SQL Server, MySQL, PostgreSQL, Oracle, and MariaDB.

Steps to Get Started with Amazon RDS - SQL Server:

1. **Access the AWS Management Console**
   - Sign in to the AWS Management Console.
   - Navigate to RDS under the "Database" section.
2. **Create a SQL Server RDS Instance**
   - Click on **Create database**.
   - **Database Creation Method:**
     - Choose either **Standard Create** for detailed setup options or **Easy Create** for basic configurations.
   - **Engine Options:**
     - Select **Microsoft SQL Server** from the list of database engines.
     - Choose the specific SQL Server edition (e.g., Express, Web, Standard, or Enterprise).
   - **Version:**
     - Select the version of SQL Server you want to use (e.g., SQL Server 2019).
   - **Instance Specifications:**
     - Choose the **DB instance class** based on your performance needs (e.g., db.t3.micro for a free-tier eligible instance or db.m5.large for higher performance).
   - **Storage:**
     - Set the allocated storage size. You can also enable **storage autoscaling** to allow the database to grow automatically as needed.
   - **Database Settings:**
     - Set the **DB Instance Identifier** (name of your database instance).
     - Define the **Master Username** and **Master Password**.
   - **VPC and Network Settings:**

Docker Compose to manage multi-container Docker applications.

- Example docker-compose.yml file:
- version: '3'
- services:
- web:
- image: my-node-app
- ports:
- - "8080:8080"
- redis:
- image: "redis:alpine"
- Start the services with:
- docker-compose up

MANAGING RESOURCES USING TERRAFORM:

Managing Resources Across Various Cloud Providers Using Terraform

Steps to Manage Resources Using Terraform:

1. Install Terraform:

- Download and install Terraform from the official website.
- Verify the installation:
- terraform --version

2. Create a Terraform Configuration File:

- Define your infrastructure in a .tf file. For example, to create an EC2 instance in AWS:
- provider "aws" {
- region = "us-west-2"
- }
- resource "aws_instance" "example" {
- ami = "ami-0c55b159cbfafe1f0"
- instance_type = "t2.micro"
- tags = {
- Name = "ExampleInstance"
- }
- }
- This example sets up an AWS provider and defines a single EC2 instance.

3. Initialize Terraform:

- Run the following command to initialize Terraform. This command downloads the required provider plugins:
- terraform init

4. Preview the Changes:

- Before applying the changes, you can preview what Terraform will do:
- terraform plan

5. Apply the Configuration:

- To create the resources defined in your configuration file, run:
- terraform apply
- Terraform will prompt you to confirm the action. Type yes to proceed.

6. Managing Resources:

- Terraform can be used to manage the entire lifecycle of your infrastructure, including updates and deletions.
- To update resources, modify the .tf file and run:
- terraform apply
- To delete resources, use:
- terraform destroy

        - Select the VPC, subnet, and security group settings that will govern network access to your instance.
    - **Backup and Maintenance:**
        - Configure automated backups, backup retention period, and the preferred backup window.
        - Set up a maintenance window if you want to specify a particular time for updates and patches.
    - Click **Create database** to start the creation process.
3. **Connect to the SQL Server RDS Instance**
    - Once the RDS instance is available (this may take a few minutes), you can connect to it using **SQL Server Management Studio (SSMS)** or another SQL client.
    - **Endpoint:**
        - Find the **endpoint and port** in the RDS console under the **Connectivity & security** tab.
        - Use these details in your SQL client to connect.
    - **Connect via SSMS:**
        - Open **SSMS**, enter the RDS **endpoint** as the Server name.
        - Choose **SQL Server Authentication** and enter the master username and password.
4. **Manage and Monitor the RDS Instance**
    - Use the **RDS Dashboard** to monitor the instance's performance, including CPU usage, memory, and storage.
    - Configure **CloudWatch Alarms** to alert you if the database goes beyond certain thresholds (e.g., high CPU usage or low available storage).
    - Perform administrative tasks like creating snapshots (manual backups), restoring backups, or scaling the instance size.
5. **Backup and Restore**
    - **Automated Backups:** Ensure automated backups are enabled. You can configure the backup retention period.

○ **Manual Snapshots:** Create a manual snapshot to capture the current state of your database. This snapshot can be used later for restoring or replicating the database. ○ **Restore a Database:** To restore, go to the **Snapshots** tab, select the snapshot, and choose **Restore Snapshot**. 6. **Security and Access Control** ○ Use **AWS IAM** to control who can manage the RDS instances. ○ Ensure **encryption** is enabled for data at rest by using AWS-managed keys. ○ Use **SSL/TLS** to secure data in transit between your application and the RDS instance. 7. **Terminate the RDS Instance** ○ When you no longer need the RDS instance, terminate it to avoid charges. ○ Go to the **RDS Dashboard**, select the instance, and click **Delete**. ○ You can choose to **retain or delete the final snapshot** during the deletion process.		