Comprehensive AWS Guide with Tutorials

Table of Contents

Introduction

Amazon Web Services (AWS) is a comprehensive cloud computing platform offering a wide range of services. This guide covers some basic AWS services, how to use them, and includes detailed tutorials for common tasks.

Useful Link:

- AWS Cloud Computing Overview

Key AWS Services

1. EC2 (Elastic Compute Cloud)

EC2 provides scalable computing capacity in the cloud, allowing you to run virtual servers for various applications.

Key Features:

- Scalable: Easily increase or decrease capacity within minutes
- Flexible: Choose from various instance types optimized for different use cases
- Cost-effective: Pay only for the compute time you use

Basic usage:

- Launch an EC2 instance:
    1. Go to EC2 dashboard
    2. Click "Launch Instance"

3. Choose an Amazon Machine Image (AMI)
4. Select instance type
5. Configure instance details
6. Add storage
7. Add tags
8. Configure security group
9. Review and launch

Advanced Features:

- Auto Scaling: Automatically adjust the number of EC2 instances based on demand
- Elastic Load Balancing: Distribute incoming traffic across multiple EC2 instances
- Elastic IP Addresses: Static IP addresses designed for dynamic cloud computing

Useful Links:

- Amazon EC2 Documentation
- EC2 Instance Types
- EC2 Pricing

2. S3 (Simple Storage Service)

S3 is an object storage service offering industry-leading scalability, data availability, security, and performance.

Key Features:

- Durability: 99.999999999% durability
- Scalability: Store and retrieve any amount of data
- Security: Comprehensive security and compliance capabilities

Basic usage:

- Create a bucket:

    1. Go to S3 dashboard
    2. Click "Create bucket"
    3. Name your bucket (must be globally unique)
    4. Choose region
    5. Set permissions
    6. Create bucket

- Upload an object:

    1. Open the bucket
    2. Click "Upload"
    3. Select files to upload
    4. Set permissions
    5. Upload

Advanced Features:

- Versioning: Maintain multiple versions of an object

- Lifecycle Policies: Automate moving objects between storage classes
- Server-Side Encryption: Automatically encrypt data at rest

Useful Links:

- Amazon S3 Documentation
- S3 Storage Classes
- S3 Pricing

3. RDS (Relational Database Service)

RDS makes it easy to set up, operate, and scale a relational database in the cloud.

Key Features:

- Automated Patching: RDS automatically patches the database software
- Backups: Automated backups with point-in-time recovery
- Multi-AZ Deployments: Enhanced availability and durability

Basic usage:

- Create a database:
    1. Go to RDS dashboard
    2. Click "Create database"
    3. Choose database engine (MySQL, PostgreSQL, Oracle, SQL Server)
    4. Select use case
    5. Specify DB details (instance specifications, storage)
    6. Configure advanced settings (network, encryption, backups)
    7. Create database

Advanced Features:

- Read Replicas: Create read-only copies of your database to offload reads
- Performance Insights: Monitor database performance and analyze issues
- Aurora: Amazon's custom built MySQL and PostgreSQL compatible database

Useful Links:

- Amazon RDS Documentation
- RDS Database Engines
- RDS Pricing

4. Lambda

Lambda lets you run code without provisioning or managing servers, providing a serverless compute service.

Key Features:

- Event-driven: Lambda functions can be automatically triggered by AWS services
- Scalability: Automatically scales your application by running code in response to each trigger
- Pay-per-use: You are charged for every 100ms your code executes and the number of times your code is triggered

Basic usage:

- Create a function:
    1. Go to Lambda dashboard
    2. Click "Create function"
    3. Choose "Author from scratch"
    4. Name your function
    5. Choose runtime (e.g., Node.js, Python, Java)
    6. Create function
    7. Write or upload your code

Advanced Features:

- Layers: Package and manage dependencies separately from your function code
- Versions and Aliases: Manage different versions of your functions
- Step Functions: Coordinate multiple Lambda functions into complex workflows

Useful Links:

- AWS Lambda Documentation
- Lambda Runtimes
- Lambda Pricing

5. IAM (Identity and Access Management)

IAM enables you to manage access to AWS services and resources securely.

Key Features:

- Fine-grained access control: Define precisely who can access which services and resources
- Multi-factor authentication (MFA): Add an extra layer of security
- Identity federation: Use existing identities from your enterprise directory

Basic usage:

- Create a user:
    1. Go to IAM dashboard
    2. Click "Users" then "Add user"
    3. Set user details
    4. Set permissions (attach policies or add to groups)
    5. Review and create

Advanced Features:

- IAM Roles: Define a set of permissions for making AWS service requests
- Policy Simulator: Test and troubleshoot IAM permissions
- Access Analyzer: Identify resources in your organization that are shared with an external entity

Useful Links:

- AWS IAM Documentation
- IAM Best Practices

- IAM Policy Examples

AWS CLI (Command Line Interface)

The AWS CLI allows you to interact with AWS services from the command line, enabling scripting of your AWS operations.

Installation:

- For Windows: Download and run the AWS CLI MSI installer
- For macOS and Linux: Use pip: `pip install awscli`

Configuration:
Run `aws configure` and provide:

- AWS Access Key ID
- AWS Secret Access Key
- Default region name
- Default output format

Example commands:

- List S3 buckets: `aws s3 ls`
- Create an S3 bucket: `aws s3 mb s3://bucket-name`
- List EC2 instances: `aws ec2 describe-instances`
- Create a new IAM user: `aws iam create-user --user-name NewUser`

Useful Links:

- AWS CLI Documentation
- AWS CLI Command Reference
- AWS CLI Configuration

Tutorials

Tutorial 1: Launching an EC2 Instance

1. **Sign in to AWS Console**: Go to the AWS Management Console and sign in.

2. **Navigate to EC2**: Click on "Services" and select "EC2" under Compute.

3. **Launch Instance**:

   - Click the "Launch Instance" button.
   - Choose an Amazon Machine Image (AMI). For this tutorial, select "Amazon Linux 2 AMI".

4. **Choose Instance Type**:

   - Select t2.micro (Free tier eligible).
   - Click "Next: Configure Instance Details".

5. **Configure Instance**:

   - Leave default settings.
   - Click "Next: Add Storage".

6. **Add Storage**:

   - The default 8 GB is sufficient for this tutorial.
   - Click "Next: Add Tags".

7. **Add Tags**:

   - Click "Add Tag".
   - Set Key as "Name" and Value as "MyFirstEC2Instance".
   - Click "Next: Configure Security Group".

8. **Configure Security Group**:

   - Create a new security group.
   - Add a rule to allow SSH (port 22) from your IP address.
   - Click "Review and Launch".

9. **Review and Launch**:

   - Review your instance configuration.
   - Click "Launch".

10. **Create Key Pair**:

    - Select "Create a new key pair".
    - Name it "MyEC2KeyPair".
    - Download the key pair and keep it safe.
    - Click "Launch Instances".

11. **Access Your Instance**:

    - Wait for the instance to start (Status Checks: 2/2 checks passed).
    - Select the instance and click "Connect".
    - Follow the instructions to SSH into your instance using your key pair.

Useful Links:

- Launch an EC2 Instance
- Connect to Your EC2 Instance

Tutorial 2: Creating and Using S3 Buckets

1. **Sign in to AWS Console**: Go to the AWS Management Console and sign in.

2. **Navigate to S3**: Click on "Services" and select "S3" under Storage.

3. **Create a Bucket**:

   - Click "Create bucket".
   - Enter a unique bucket name, e.g., "my-first-s3-bucket-[yourname]".

    - Choose a region close to you.
    - Leave other settings as default.
    - Click "Create bucket".

4. **Upload an Object**:

    - Click on your new bucket name.
    - Click "Upload".
    - Click "Add files" and select a file from your computer.
    - Click "Upload".

5. **Set Permissions**:

    - After upload, click on the file name.
    - Go to the "Permissions" tab.
    - To make the file public, click "Edit" under "Public access" and select "This object has public access".
    - Save changes.

6. **Access Your File**:

    - In the "Object overview" section, you'll find the Object URL.
    - Copy this URL and paste it into a web browser to access your file.

7. **Create a Folder**:

    - Go back to your bucket.
    - Click "Create folder".
    - Name your folder and click "Create folder".

8. **Enable Versioning**:

    - Go to your bucket's "Properties" tab.
    - Find "Bucket Versioning" and click "Edit".
    - Enable versioning and save changes.

9. **Use AWS CLI with S3**:

    - List buckets: `aws s3 ls`
    - Upload a file: `aws s3 cp localfile.txt s3://your-bucket-name/`
    - Download a file: `aws s3 cp s3://your-bucket-name/file.txt ./`

Remember to delete any resources you created to avoid unexpected charges.

Useful Links:

- Create an S3 Bucket
- Upload Objects to S3

Tutorial 3: Setting Up a Lambda Function

1. **Sign in to AWS Console**: Go to the AWS Management Console and sign in.

2. **Navigate to Lambda**: Click on "Services" and select "Lambda" under Compute.

3. **Create Function**:

   - Click "Create function".
   - Choose "Author from scratch".
   - Enter a function name, e.g., "MyFirstLambdaFunction".
   - For Runtime, choose "Python 3.8" (or your preferred language).
   - For Execution role, choose "Create a new role with basic Lambda permissions".
   - Click "Create function".

4. **Write Function Code**:

   - In the "Function code" section, you'll see a simple Python function.

   - Replace the code with the following:

     ```
     import json

     def lambda_handler(event, context):
         name = event.get('name', 'World')
         return {
             'statusCode': 200,
             'body': json.dumps(f'Hello, {name}!')
         }
     ```

   - Click "Deploy" to save your changes.

5. **Test Your Function**:

   - Click "Test" near the top of the page.

   - Create a new test event. Name it "TestEvent".

   - Replace the default JSON with:

     ```
     {
       "name": "Alice"
     }
     ```

   - Click "Create".

   - Click "Test" again to run your function with this test event.

   - You should see the execution result with "Hello, Alice!" in the response.

6. **Create an API Gateway Trigger**:

   - In the "Designer" section, click "Add trigger".
   - Choose "API Gateway" from the list.

    - For API, select "Create an API".
    - Choose "REST API" and "Open" security.
    - Click "Add".

7. **Test Your API**:

    - After the API Gateway is added, click on it in the Designer.
    - In the API Gateway section, you'll find an API endpoint URL.
    - Copy this URL and paste it into a new browser tab.
    - You should see the response: "Hello, World!"

8. **Customize Your API**:

    - To pass a name parameter, append `?name=YourName` to the URL.
    - For example: `https://your-api-id.execute-api.region.amazonaws.com/default/MyFirstLambdaFunction?name=Alice`
    - This should return: "Hello, Alice!"

Useful Links:

- Create a Lambda Function
- Using Lambda with API Gateway

Best Practices

1. Always use IAM users instead of the root account for better security
2. Implement Multi-Factor Authentication (MFA) for all users
3. Follow the principle of least privilege when assigning permissions
4. Regularly review and audit your AWS resources and permissions