**Introduction to AWS CLI**

**Introduction**

In today’s era of cloud computing, managing cloud resources efficiently is essential. Amazon Web Services (AWS) is a prominent player in the cloud computing market, providing a wide range of services and features. To interact with AWS services via the command line, AWS CLI (Command Line Interface) offers a powerful and efficient solution. In this blog post, we will take you from a beginner level to a pro in AWS CLI, covering all the essential aspects along the way.

**What is AWS CLI?**

AWS CLI is a unified tool developed by Amazon that enables users to interact with AWS services through the command line. It provides a command line interface for performing various administrative tasks, automating operations, and managing AWS resources. AWS CLI offers a consistent experience across different AWS services and regions, making it a versatile tool for developers, system administrators, and DevOps engineers.

**Installation and Configuration**

To begin using AWS CLI, you need to install it on your local machine:

1. Install AWS CLI:

* **For Windows:**Download the MSI installer from the AWS Command Line Interface official documentation and follow the installation wizard.
* **For macOS:** Install AWS CLI using Homebrew by running **brew install awscli** in the Terminal.
* **For Linux:**Follow the instructions provided in the AWS Command Line Interface official documentation for your specific distribution.

2. Open a terminal or command prompt.

3. Run the following command to start the configuration process:

**aws configure**

4. Enter your AWS Access Key ID and Secret Access Key when prompted. These credentials are associated with an IAM user that has appropriate permissions to access the AWS resources you intend to manage using AWS CLI.

5. Provide the default AWS region you want to use for your commands. For example, enter us-east-1 for the US East (N. Virginia) region.

6. Specify the output format for the AWS CLI commands. The default format is JSON, but you can choose from several options, including JSON, text, table, or YAML.

7. Press Enter to complete the configuration.

8. AWS CLI will store the provided configuration in a file located **~/.aws/credentials** on macOS and Linux, or **C:\Users\USERNAME\.aws\credentials** on Windows. The configuration includes the Access Key ID, Secret Access Key, default region, and output format.

9. Optionally, you can also configure additional settings such as named profiles, which allow you to manage multiple sets of AWS credentials. To set up a named profile, you can run the aws configure command again and provide a unique profile name when prompted.

That’s it! You have successfully configured AWS CLI with your credentials and default settings. You can now start using AWS CLI commands to interact with various AWS services from the command line.

Remember, it’s important to keep your AWS credentials secure and avoid sharing them. If you need to rotate or update your credentials, you can re-run the **aws configure** command and provide the new credentials.

**Basic Usage and Command Structure**

AWS CLI operates using a set of commands structured in a hierarchical manner. The command structure follows the pattern: **aws <service> <command> <subcommand>.** I'll explain each component in detail and demonstrate how to use AWS CLI commands to perform tasks like creating S3 buckets, launching EC2 instances, managing IAM users, and more.

**Working with AWS Services**

AWS CLI provides comprehensive support for interacting with a vast array of AWS services. I’ll introduce you to some of the core services and demonstrate how to use AWS CLI to perform common tasks, such as managing S3 objects, launching EC2 instances, configuring auto-scaling groups, managing, and more. By the end, you’ll have a good understanding of how to leverage AWS CLI for various use cases.

**Create Your First Key Pair & EC2 instance using AWS CLI**

To create a key pair using AWS CLI, you can use the **create-key-pair** command. Here's an explanation of how it works and why it is needed:

**Create a key pair**

**What is a Key Pair?**

A key pair is a secure way to connect to your Amazon EC2 instances. It consists of a public key and a private key. The public key is stored on the EC2 instance and allows you to authenticate and securely access the instance, while the private key is kept securely on your local machine and is used to prove your identity.

1. Creating a Key Pair with AWS CLI:

To create a key pair using AWS CLI, you can use the **create-key-pair** command. Here's the basic syntax:

**aws ec2 create-key-pair --key-name <KEY\_PAIR\_NAME> --query 'KeyMaterial' --output text > <KEY\_PAIR\_FILE\_NAME>.pem**

Let’s break down the command and its parameters:

* **--key-name:** Specify a unique name for the key pair.
* **--query 'KeyMaterial' --output text:** Retrieve only the value of the key material (the actual private key) from the response.
* **> <KEY\_PAIR\_FILE\_NAME>.pem:** Redirect the key material output to a file with the specified name and a **.pem** file extension. This file will contain the private key.

2. Replace the **<KEY\_PAIR\_NAME**> with a meaningful name for your key pair, and **<KEY\_PAIR\_FILE\_NAME>** with the desired name for the file that will store the private key.

3. Here’s an example command to create a key pair named “my-key-pair” and store the private key in a file named “my-key-pair.pem”:

**aws ec2 create-key-pair --key-name my-key-pair --query 'KeyMaterial' --output text > my-key-pair.pem**

4. After executing the command, AWS CLI will create the key pair and return the private key material as a response. The private key is saved in the specified file.

5. Keep the private key file (<KEY\_PAIR\_FILE\_NAME>.pem) secure and accessible only to authorized individuals. You will need this private key when connecting to your EC2 instances.

Key pairs are essential because they provide a secure method to authenticate and access your EC2 instances. When launching an EC2 instance, you can specify the key pair name, and AWS will associate the public key with the instance. This allows you to connect to the instance securely using SSH or other remote access methods.

By creating a key pair with AWS CLI, you have the flexibility to generate and manage key pairs programmatically, making it easier to automate the deployment and management of EC2 instances in your infrastructure.

To check the existing key pairs in your AWS account using AWS CLI, you can use the **describe-key-pairs**command. Here's how you can use it:

1. Open a terminal or command prompt.
2. Run the following command:

**aws ec2 describe-key-pairs**

3. This command will retrieve information about all the key pairs in your AWS account.

4. The response will include details such as the key pair name, key pair ID, and fingerprint for each key pair.

**To delete a key pair using AWS CLI, you can use the**delete-key-pair**command. Here's how you can do it**

1. Open a terminal or command prompt.
2. Run the following command:

**aws ec2 delete-key-pair --key-name <KEY\_PAIR\_NAME>**

Replace <KEY\_PAIR\_NAME> with the name of the key pair, you want to delete.

3. After executing the command, AWS CLI will attempt to delete the specified key pair.

4. If the key pair is successfully deleted, the command will not produce any output.

**Create your first EC2 instance**

1. First, make sure you have AWS CLI installed and configured with the appropriate credentials.

To create an EC2 instance, you can use the run-instances command. Here's the basic syntax:

**aws ec2 run-instances --image-id <AMI\_ID> --instance-type <INSTANCE\_TYPE> --key-name <KEY\_PAIR\_NAME> --security-group-ids <SECURITY\_GROUP\_ID> --subnet-id <SUBNET\_ID> --count <INSTANCE\_COUNT>**

**Let’s break down the command and its parameters**

* ***--image-id:*** Specify the ID of the Amazon Machine Image (AMI) you want to use for the instance. This determines the operating system and software pre-installed on the instance.
* ***--instance-type:*** Select the instance type, which defines the hardware of the host computer used for the instance.
* ***--key-name:*** Provide the name of the Key Pair used to connect to the instance securely.
* ***--security-group-ids:*** Specify the ID of the security group that controls inbound and outbound traffic for the instance.
* ***--subnet-id:*** Set the ID of the subnet in which the instance will be launched.
* ***--count:*** Determine the number of instances to launch.

3. Replace the placeholders (<AMI\_ID>, <INSTANCE\_TYPE>, <KEY\_PAIR\_NAME>, <SECURITY\_GROUP\_ID>, <SUBNET\_ID>, <INSTANCE\_COUNT>) with the actual values according to your requirements.

4. Here’s an example command that creates a single t2.micro EC2 instance in the default VPC:

**aws ec2 run-instances --image-id ami-12345678 --instance-type t2.micro --key-name my-key-pair --security-group-ids sg-12345678 --subnet-id subnet-12345678 --count 1**

5. After executing the command, AWS CLI will return a response containing information about the created instance, such as the instance ID, public IP address, and more.

Remember to adjust the command parameters based on your specific needs, such as the AMI, instance type, key pair, security group, subnet, and instance count.

By mastering the ***run-instances*** command and understanding its parameters, you can easily create EC2 instances programmatically and integrate them into your automation workflows.

**How can you Tag, List, and Terminate an EC2 Instance**

Now quickly I’ll show you how you can tag, list, and terminate your ec2 instance. Follow the below commands:

1. Tagging an EC2 Instance:

***aws ec2 create-tags --resources <INSTANCE\_ID> --tags Key=<TAG\_KEY>,Value=<TAG\_VALUE>***

2. Listing your Instances:

***aws ec2 describe-instances***

3. Terminating an Instance:

***aws ec2 terminate-instances --instance-ids <INSTANCE\_ID>***

Hopefully, you’ll able to perform these commands to tag, list, and terminate your instances by reading this full post.