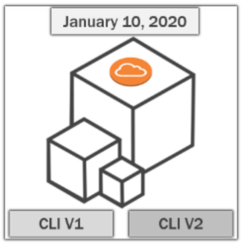
AWS Command Line Interface - AWS CLI

What is AWS CLI?

The AWS Command Line Interface (AWS CLI) is an open-source tool that allows you to use commands in your command-line shell to interact with AWS services.

- The AWS CLI controls the AWS resources on your own device from a terminal session, which allows you to control, monitor and configure multiple AWS services.
- The AWS CLI allows you to start running commands with a minimal configuration that enforce functionality similar to that offered by the browser-based AWS Management Console from the command prompt in your favorite terminal program: Linux, Mac, or Windows.

AWS CLI - Versions



On January 10th, 2020, AWS CLI version 1, which requires a separate installation of Python to operate, stopped supporting Python versions 2.6 and 3.3. All builds of AWS CLI version 1 released after January 10th, 2020, starting with version 1.17, require Python 2.7, Python 3.4, or a later version to successfully use the AWS CLI.

AWS CLI version 1

AWS CLI version 1 is the original AWS CLI, and AWS continues to support it. However, major new features that are introduced in AWS CLI version 2 might not be backported to AWS CLI version 1. To use those features, you must install AWS CLI version 2.

AWS CLI version 2

AWS CLI version 2 is the most recent major version of the AWS CLI and supports all of the latest features. Some features introduced in version 2 are not backward compatible with version 1 and you must upgrade to access those features.

Migrating from AWS CLI version 1 to version 2

If you have used and ran commands or scripts with AWS CLI version 1 before and you are considering migrating to AWS CLI version 2, see Breaking Changes – Migrating from AWS CLI version 1 to version 2 for a description of the changes that AWS recommends you to know about.

Migrating from AWS CLI version 1 to version 2

If you have used and ran commands or scripts with AWS CLI version 1 before and you are considering migrating to AWS CLI version 2, see Breaking Changes – Migrating from AWS CLI version 1 to version 2 for a description of the changes that AWS recommends you to know about.

⚠**Caution ! :**

- AWS CLI version 2 is provided as a preview for testing and evaluation. At this time, AWS does not recommend using it in a production environment. For production environments, AWS recommends that you use the generally available version 1.
- However, you can have the opportunity to experience the newest features by using CLI as version 2 in the training applications we will apply during the course.

AWS CLI - Introduction

After you have finished installing AWS CLI, let's check it and set up AWS configuration.

- Open your terminal.
- Type `aws` command and click ENTER.

```
elon@PC:~$ aws
usage: aws [options]   [ ...] [parameters]
To see help text, you can run:

  aws help
  aws  help
  aws   help
aws: error: the following arguments are required: command
```

- The **aws** command was detected by the terminal, but a response was returned indicating deficiencies in its usage and what we should do to get help.
- Verify the setup by typing the `aws help` command.

```
elon@PC:~$ aws help

```

- A list of valid AWS commands should appear in the AWS CLI window.
- Type `aws --version` command to see current version of installed aws cli.

```
elon@PC:~$ aws --version
aws-cli/1.14.44 Python/3.6.9 Linux/4.4.0-18362-Microsoft botocore/1.8.48

```

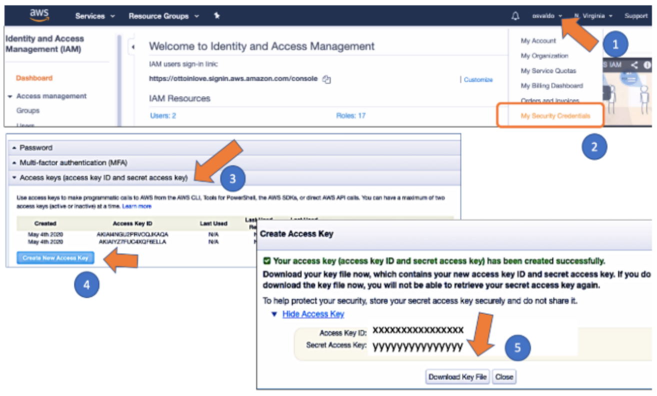
- After making sure that AWS CLI is installed successfully, now let's move on to the configuration section.

AWS CLI - Configuration

While installing a CLI, first the system requires you to enter your **Access Key ID** and **Secret Access Key.**

As you remember from the "Creating a User" process, AWS gives you **Access Key ID Key** and **Secret Access Key** via credentials.csv file if you select the "Programmatic Access" option. And, you can either download credentials.csv file in your computer or send e-mail to your account.

You can also reach Access Key ID from "**My Security Credentials**". But AWS, for your own security, **does not allow retrieval of a "Secret Access Key" after its initial creation.** This applies to both root secret access keys and Identity and Access Management (IAM) user Secret Access Keys. As a security best practice, you should securely store your secret access keys.



Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, the AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

Created	Access Key ID	Last Used
May 4th 2020	AKIAI4NGU2PRVOQJKAQA	N/A
May 4th 2020	AKIAIYZ7FUC4XQF6ELLA	N/A

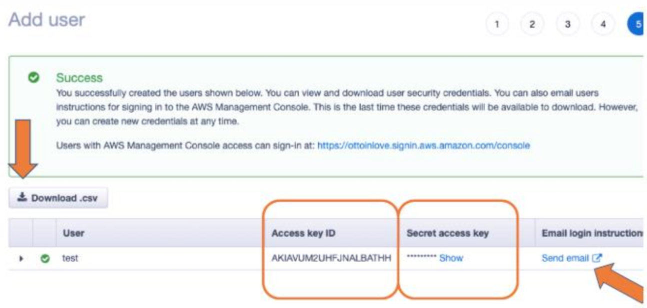
Your access key (access key ID and secret access key) has been created successfully.

Download your key file now, which contains your new access key ID and secret access key. If you do download the key file now, you will not be able to retrieve your secret access key again.

To help protect your security, store your secret access key securely and do not share it.

Access Key ID: XXXXXXXXXXXXXXXX
Secret Access Key: YYYYYYYYYYYYYYYY

If you have lost your secret access key, you can always replace it by creating a new access key from the "**My Security Credentials**" tab as you see in the picture above.



Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://ottoinlove.signin.aws.amazon.com/console

User	Access key ID	Secret access key	Email login instruction
test	AKIAVUM2UHFJNALBATHH	********* Show	Send email

You may also create a new IAM user and then you can save the credentials.csv file at the end of the user creation process.

The **aws configure** command is the fastest way to set up your AWS CLI installation.

- Open your terminal.
- Type `aws configure` command and click ENTER.

```
elon@PC:~$ aws configure
```

- AWS CLI prompts you for four pieces of information (access key, secret access key, AWS Region, and output format) respectively.
- You are asked to enter **AWS Access Key ID** first.
- Type your **AWS Access Key ID** and click ENTER.

```
AWS Access Key ID [****************]:

```

- Then, you are asked to enter **AWS Secret Access Key**.
- Type your **AWS Secret Access Key** and click ENTER.

```
AWS Secret Access Key [****************exit]:

```

- Then, you are asked to enter **Default region name**.
- **Default region name** identifies the AWS Region whose servers you want to send your requests to by default. This is typically the Region closest to you, but it can be any Region. For example, you can type `us-west-2` to use US West (Oregon) or `us-west-1` to use US East (N.Virginia). This is the Region that all later requests are sent to, unless you specify otherwise in an individual command.
- Type **Default region name** and click ENTER.

```
Default region name [q]:

```

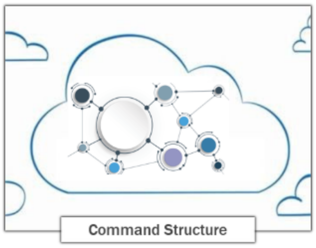
- Then, you are asked to enter Default output format.
- The `Default output format` specifies how the results are formatted. The value can be any of the values in the following list: json, yaml, text, table.
- If you don't specify an output format, `json` is used as the default.
- Type **Default output format** which format you prefer and click ENTER.

```
Default output format [json:]:

```

- Now, all of the four steps of the configuration process are done as below.

```
AWS Access Key ID [***********]: AKIXXXXXXXXXXXXXXRVEO
AWS Secret Access Key [***********]: L91XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXSw/
Default region name [us-west-1]: us-west-1
Default output format [json:]: json

```

AWS CLI - Command Structure



AWS CLI is a management interface that allows us to do everything we do via the management console by entering the commands.

- Knowing and learning this really well will make our life much easier.
- As we dive deep into the AWS world, we will see that this is like a huge sea that will grow day by day as we work on the environment. Therefore, it will become almost impossible to manage manually after a while.
- Therefore, it is important to understand the command structure well.
- It is not possible to learn this structure completely at once. As we use the AWS CLI and become familiar with AWS services, it will evolve in our competence on AWS CLI commands.

After configuring AWS on the terminal, we are now connected to the AWS world. The AWS Command Line Interface (AWS CLI) uses a multipart structure on the command line that must be specified in this order:

```
$ aws <command> <subcommand> [options and parameters]
```

- The base call to the `aws` program.
- The top-level `commands` , which typically corresponds to an **AWS service** supported by the AWS CLI.
- The `subcommands` *subcommand* that specifies which operation to perform.
- General CLI `options` or `parameters` required by the operation. You can specify these in any order as long as they follow the first three parts. If an exclusive parameter is specified multiple times, only the *last value* applies.

AWS CLI - Example

Let's use some of AWS CLI commands that perform common tasks to understand the command structure better by getting our hands dirty with AWS CLI.

- `aws` is the beginning of all commands, it is mandatory to write this.
- With the `aws help` command, we can get help from AWS about the commands we can use in AWS.

```
$ aws help
```

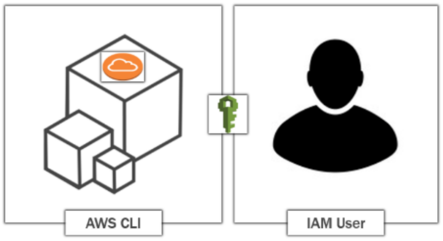
If we use the `aws iam help` command, we can only see the CLI information, parameters, usage details for the iam service.

```
$ aws iam help
```

If we use the `aws iam ls help` command, we see how to use this ls command.

```
$ aws iam ls help
```

AWS CLI - Creating IAM User



Let's see how to create an IAM user via using CLI command generally.

- Begin with `aws` command as it is the beginning of all commands.
- Specify the AWS service that you want to work with. Because we want to create an IAM user, this command is `iam` in this case.
- Then, use `create-user` subcommand and `--user-name` parameter to specify user-name.
- Finally type the user-name you want to give to the new user, for example, MyUser.

```
$ aws iam create-user --user-name MyUser
```

Now, let's create a new user using AWS CLI.

- First, list the users currently on our AWS account.
- Then create a new user.
- Finally, check the process of creating a new user by re-listing the users.

Listing IAM users:

```
$ aws iam list-users
```

- After running this command, the terminal will return to us the existing users in our AWS account with some identifying information as below.

Listing IAM users:

```
$ aws iam list-users
```

- After running this command, the terminal will return to us the existing users in our AWS account with some identifying information as below.

```
{
    "Users": [
        {
            "Path": "/",
            "UserName": "Callahan",
            "UserId": "AIDAY7KQYXISSEOT3CBI7",
            "Arn": "arn:aws:iam::617034070565:user/Callahan",
            "CreateDate": "2020-02-02T01:33:06Z"
        },
        {
            "Path": "/",
            "UserName": "ElonAdmin",
            "UserId": "AIDAY7KQYXISRS4QSCBXJ",
            "Arn": "arn:aws:iam::617034070565:user/ElonAdmin",
            "CreateDate": "2020-01-28T22:40:48Z",
            "PasswordLastUsed": "2020-02-04T05:51:41Z"
        },
        {
            "Path": "/",
            "UserName": "Eric",
            "UserId": "AIDAY7KQYXISWL3DE5V6D",
            "Arn": "arn:aws:iam::617034070565:user/Eric",
            "CreateDate": "2020-02-02T01:32:07Z"
        },
        {
            "Path": "/",
            "UserName": "Mary",
            "UserId": "AIDAY7KQYXISUCYE3Z66C",
            "Arn": "arn:aws:iam::617034070565:user/Mary",
            "CreateDate": "2020-02-02T01:32:43Z"
        },
        {
            "Path": "/",
            "UserName": "Osvaldo",
            "UserId": "AIDAY7KQYXISU7RJFZCTA",
            "Arn": "arn:aws:iam::617034070565:user/Osvaldo",
            "CreateDate": "2020-02-04T08:56:05Z"
        }
    ]
}
```

- As you can see, we have currently five IAM users who are: Callahan, ElonAdmin, Eric, Mary, Osvaldo.

Creating a New user:

- Type the needed command described above with the user-name of John.

```
$ aws iam create-user --user-name John
```

- After running this command, the terminal will return to us the newly created user with some identifying information.

```
$ {
    "User": {
        "Path": "/",
        "UserName": "John",
        "UserId": "AIDAY7KQYXISTFCNOE77M",
        "Arn": "arn:aws:iam::617034070565:user/John",
        "CreateDate": "2020-02-04T09:19:38Z"
    }
}
```

Re-listing IAM users:

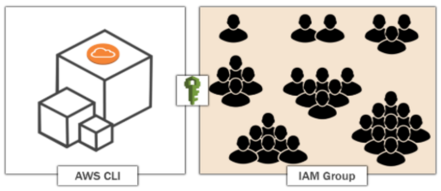
```
$ aws iam list-users
```

When we list the IAM users with this command again, if we have successfully created the user, the user named John should also be listed as follows.

```
$ {
    "Users": [
        {
            "Path": "/",
            "UserName": "Callahan",
            "UserId": "AIDAY7KQYXISSEOT3CBI7",
            "Arn": "arn:aws:iam::617034070565:user/Callahan",
            "CreateDate": "2020-02-02T01:33:06Z"
        },
        {
            "Path": "/",
            "UserName": "ElonAdmin",
            "UserId": "AIDAY7KQYXISRS4QSCBXJ",
            "Arn": "arn:aws:iam::617034070565:user/ElonAdmin",
            "CreateDate": "2020-01-28T22:40:48Z",
            "PasswordLastUsed": "2020-02-04T05:51:41Z"
        },
        {
            "Path": "/",
            "UserName": "Eric",
            "UserId": "AIDAY7KQYXISWL3DE5V6D",
            "Arn": "arn:aws:iam::617034070565:user/Eric",
            "CreateDate": "2020-02-02T01:32:07Z"
        },
        {
            "Path": "/",
            "UserName": "John",
            "UserId": "AIDAY7KQYXISTFCNOE77M",
            "Arn": "arn:aws:iam::617034070565:user/John",
            "CreateDate": "2020-02-04T09:19:38Z"
        },
        {
            "Path": "/",
            "UserName": "Mary",
            "UserId": "AIDAY7KQYXISUCYE3Z66C",
            "Arn": "arn:aws:iam::617034070565:user/Mary",
            "CreateDate": "2020-02-02T01:32:43Z"
        },
        {
            "Path": "/",
            "UserName": "Osvaldo",
            "UserId": "AIDAY7KQYXISU7RJFZCTA",
            "Arn": "arn:aws:iam::617034070565:user/Osvaldo",
            "CreateDate": "2020-02-04T08:56:05Z"
        }
    ]
}
```

Congratulations! The user creation process using AWS CLI has been successfully completed.

AWS CLI - Creating IAM Group



Let's see how to create an IAM group via using CLI command generally.

- Begin with `aws` command as it is the beginning of all commands.
- Specify the AWS service that you want to work with. Because we want to create an IAM group, this command is `iam` in this case.
- Then, use `create-group` subcommand and `--group-name` parameter to specify user-name.
- Finally type the user-name you want to give to the new user, for example, MyIamgroup.

```
$ aws iam create-group --group-name MyIamGroup
```

Now, let's create a new group using AWS CLI.

- First, list the groups currently on our AWS account.
- Then create a new group.
- Finally, check the process of creating a new group by re-listing the groups.

Listing IAM users:

```
$ aws iam list-groups
```

- After running this command, the terminal will return to us the existing users in our AWS account with some identifying information as below.

```
{
    "Groups": [
        {
            "Path": "/",
            "GroupName": "Admins",
            "GroupId": "AGPAY7KQYXISVILJ7OEPE",
            "Arn": "arn:aws:iam::617034070565:group/Admins",
            "CreateDate": "2020-02-02T00:59:47Z"
        },
        {
            "Path": "/",
            "GroupName": "Instructors",
            "GroupId": "AGPAY7KQYXIS2TZKDAGSB",
            "Arn": "arn:aws:iam::617034070565:group/Instructors",
            "CreateDate": "2020-02-02T01:22:48Z"
        },
        {
            "Path": "/",
            "GroupName": "Mentors",
            "GroupId": "AGPAY7KQYXISSIGZS55PL",
            "Arn": "arn:aws:iam::617034070565:group/Mentors",
            "CreateDate": "2020-02-02T01:25:12Z"
        }
    ]
}
```

- As you can see, we have currently three IAM groups which are: Admins, Instructors, Mentors.

Creating a New group:

- Type the needed command described above with the group-name of **TestCLI.**

```
$ aws iam create-group --group-name TestCLI
```

- After running this command, the terminal will return to us the newly created group with some identifying information.

```
{
    "Group": {
        "Path": "/",
        "GroupName": "TestCLI",
        "GroupId": "AGPAY7KQYXIS3UYR6KEJJ",
        "Arn": "arn:aws:iam::617034070565:group/TestCLI",
        "CreateDate": "2020-02-04T09:40:39Z"
    }
}
```

Re-listing IAM Groups:

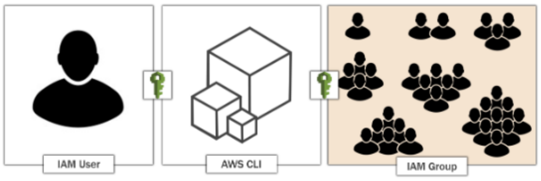
```
$ aws iam list-groups
```

When we list the IAM groups with this command again, if we have successfully created the testCLI group, it should also be listed as follows.

```
{
    "Groups": [
        {
            "Path": "/",
            "GroupName": "Admins",
            "GroupId": "AGPAY7KQYXISVILJ7OEPE",
            "Arn": "arn:aws:iam::617034070565:group/Admins",
            "CreateDate": "2020-02-02T00:59:47Z"
        },
        {
            "Path": "/",
            "GroupName": "Instructors",
            "GroupId": "AGPAY7KQYXIS2TZKDAGSB",
            "Arn": "arn:aws:iam::617034070565:group/Instructors",
            "CreateDate": "2020-02-02T01:22:48Z"
        },
        {
            "Path": "/",
            "GroupName": "Mentors",
            "GroupId": "AGPAY7KQYXISSIGZS55PL",
            "Arn": "arn:aws:iam::617034070565:group/Mentors",
            "CreateDate": "2020-02-02T01:25:12Z"
        },
        {
            "Path": "/",
            "GroupName": "TestCLI",
            "GroupId": "AGPAY7KQYXIS3UYR6KEJJ",
            "Arn": "arn:aws:iam::617034070565:group/TestCLI",
            "CreateDate": "2020-02-04T09:40:39Z"
        }
    ]
}
```

Congratulations! The group creation process using AWS CLI has been successfully completed.

AWS CLI - Adding Users to IAM Groups



Now we will practice how to add the newly created user to a group.

- Begin with `aws` command.
- Specify the AWS service as `iam` .
- Use `add-user-to-group` subcommand and `--user-name` parameter and the **value (John)** to specify which user to add to the group.
- Use `--group-name` parameter and **value (TestCLI)** to specify which group the user will be added.

```
$ aws iam add-user-to-group --user-name John --group-name TestCLI
```

- After running this command, the terminal will return no output, but AWS CLI will add an IAM user named John to the IAM group named TestCLI.

What is IAM?

AWS IAM stands for Identity & Access Management and is the primary service that handles authentication and authorization processes within AWS environments. As an AWS account management service, it lets you to control access to AWS services in a secure manner, and helps to monitor who is authenticated and allowed to use resources.

What are IAM Workflow components?

- **Principal**
- **Authentication**
- **Request**
- **Authorization**
- **Actions or Operations**
- **Resources**

What are IAM Identities?

- Users
- Groups
- Roles

What are the different types of Security Credentials in AWS?

- Root User Credentials : E-mail & Password
- IAM Uses Credentials : Username & Password
- MFA : MultiFactor Authentication via third party applications
- CLI, SDKs, API Credentials : Access Keys
- Amazon EC2 Credentials : Key Pairs

What are different IAM user types?

- Real Person
- Web Application
- Service Accounts
- Software

What are the components of 5-step security system in IAM?

- Delete your root access keys
- Activate MFA on your root account
- Create individual IAM users
- Use groups to assign permissions
- Apply an IAM password policy

What are IAM Policy Types?

- AWS managed policies (Managed policies)
- Customer managed policies (Managed policies)
- Inline policies
- Jop Function policies

What is an IAM Role?

An IAM Role is the authorization system that we determine how and with which authorizations that other AWS resources, users coming from other AWS accounts or accounts that will have access to our system as a result of the trust relationship we have established with an identity provider in the outside world can access our own AWS resources.