Professional Skills
Agile Fundamentals
Jira
Git
Databases Introduction
Java Beginner
Maven
Testing (Foundation)
Java Intermediate
HTML
CSS
Javascript
Spring Boot
Selenium
Sonarqube
Advanced Testing (Theory)
Cucumber
MongoDB
Express
NodeJS
React
Express-Testing
Networking
Security
Cloud Fundamentals
AWS Foundations
- AWS Introductions
- AWS Sign up
- AWS Billing Alert
- AWS EC2

AWS Programmatic Access

Contents

Overview

Programmatic Access is a method of invoking actions within AWS. When creating your infrastructure within the AWS environment, it is important to know who has access to your AWS resources. Programmatic access can be dangerous if not handled carefully, as third-party applications can make use of your credentials and can cause devastating impact if a malicious user gets a hold of these credentials.

Protecting your AWS programmatic access and knowing who has access falls under the AWS customer's responsibility, as well as the engineers working for an AWS customer.

You should never provide details to the root account; this is the account that you created your AWS account with. This account is, by default, the root account and therefore has unlimited access and no restrictions within your AWS environment.

You will need to create a user and give them permissions that are restricted to limited resources in AWS, and only give this user programmatic access.

You will need to decide when it is best to use programmatic access.

Applications or infrastructure that require AWS resources will need to have the credentials of the users that have programmatic access.

An example of this is building a security camera. You want to ensure the videos being recorded are uploaded to a safe location to access later; using AWS programmatic access, you can provide the security camera with access to the AWS environment to upload the videos.

Credentials

There are various ways you can use your AWS credentials in your application and scripts. When configuring your credentials on a server, a file in ~/.aws/credentials is created. This file contains the AWS access key and Secret Access Key.

Source AWS Credentials

If an application you are developing requires AWS resources, you can simply:

```
source ~/.aws/credentials
```

This command allows you to use the variables 'aws_access_key_id' and 'aws_secret_access_key' within your application and run aws commands to utilise the AWS resources.

Use prebuilt commands

If you are using an application that has a lot of support for AWS, for example Terraform, there are predefined support:

```
shared_credentials_file = "/path/to/aws/credentials/file"
```

You will need to look up the documentation regarding other applications using AWS credentials.

Exporting as Environment Variables

Another alternative is to create an environment variable so that all applications have access to your AWS credentials. This can be done by copying the value of the access key and secret access key. For example:

```
export AWS_ACCESS_KEY_ID=AKIA2W............
export AWS_SECRET_ACCESS_KEY=1hmD9........
```

Credential Profiles

Within the ~/.aws/credentials file, you will find the default profile with the access key and secret access key, which was provided when initially created. This is under the "[default]" tag.

You add additional accounts with programmatic access to the same user. You will then have to manually go into the ~/.aws/credentials file and add another hostname and its credentials. The file should look like the following:

```
[default]
aws_access_key_id=XXXXXXXXXXX
aws_secret_access_key=XXXXXXXXXXXXX

[example_1]
aws_access_key_id=YYYYYYYYYYYYY
aws_secret_access_key=YYYYYYYYYYYYYY
```

The way you would utilise the second account is by defining the profile you want to use. You can even define a different configuration for each profile. Navigating to ~/.aws/config, you will see something similar to:

```
[default]
region = eu-west-2
output = json
```

You can define a different region and output for different profiles; in this case we can configure different configuration for different profiles. For example:

```
[default]
region = eu-west-2
output = json

[profile example_1]
region = eu-west-3
output = text
```

AWS CLI

You can define which profile you want to use to run AWS commands. If none are provided, it will use the default account.

aws s3 ls --profile example_1

Use prebuilt commands

You can use an application that supports AWS to define the profile you want to use within your AWS credentials file.

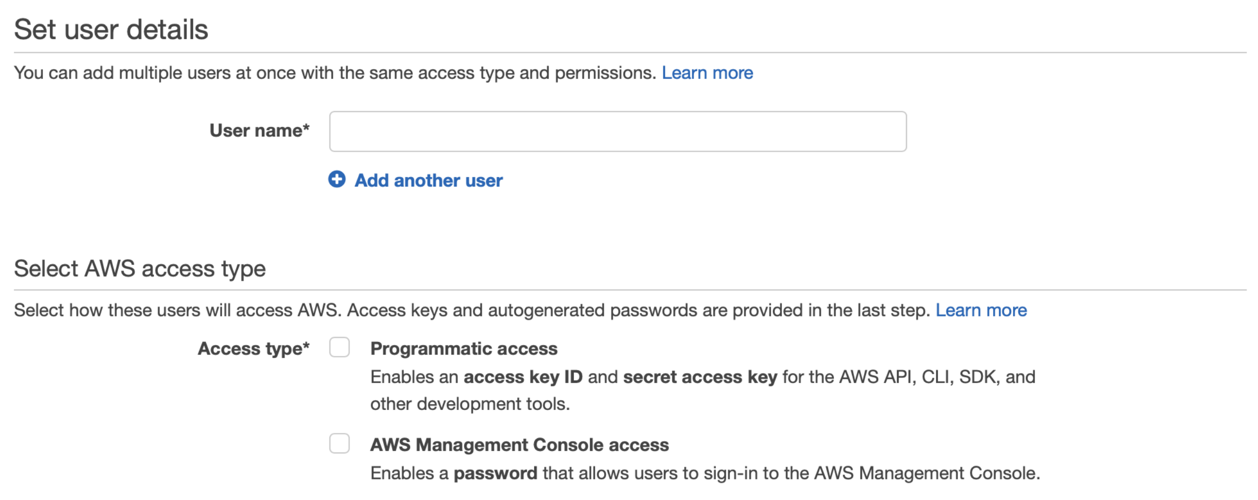
For example, Terraform can use shared_credentials_file to provide a path to where your credentials are located; if there are multiple profiles, you can define which profile to use:

```
provider "aws" {
  shared_credentials_file = "/path/to/aws/credentials/file"
  profile                 = "example_1"
}
```

This example is only for Terraform; you will need to look into the documentation for other applications that support AWS on how to use profiles.

Tutorial

1. Navigate to the AWS Console and sign in here
2. Navigate to IAM service
3. On the left go to "Users"
4. If you are new to AWS, you shouldn't see any users that have been created before. Click on "Add user" button.
5. You should see the following screen:



Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name*

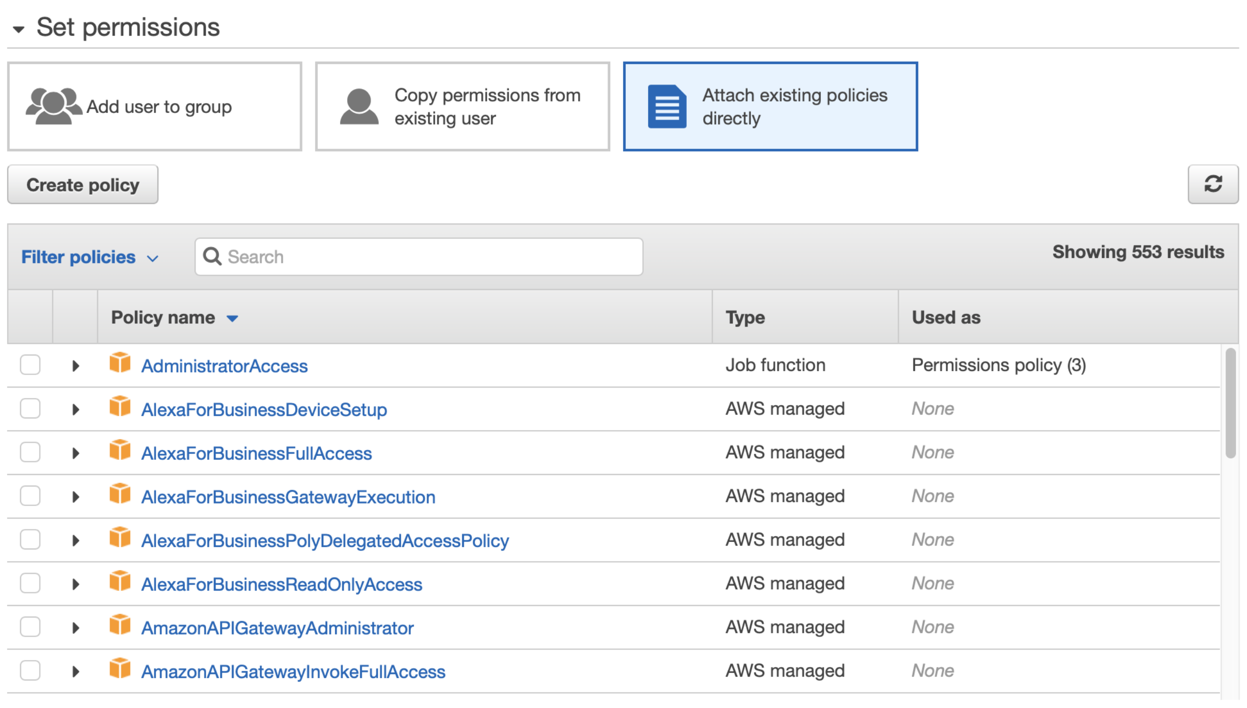
Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type*
- **Programmatic access**
  Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- **AWS Management Console access**
  Enables a **password** that allows users to sign-in to the AWS Management Console.

6. Provide a username for this user you are creating. Make sure the name is meaningful to help you keep track of all the users being created.
7. Under the "Select AWS access type", check "Programmatic Access".
8. Proceed to the next page, where you will be prompted to provide permissions you want this user to have.


Set permissions

Add user to group | Copy permissions from existing user | Attach existing policies directly

Create policy

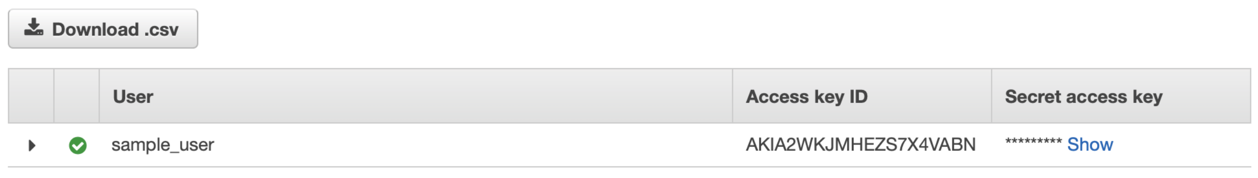
Filter policies | Search | Showing 553 results

		Policy name	Type	Used as
	▸	AdministratorAccess	Job function	Permissions policy (3)
	▸	AlexaForBusinessDeviceSetup	AWS managed	None
	▸	AlexaForBusinessFullAccess	AWS managed	None
	▸	AlexaForBusinessGatewayExecution	AWS managed	None
	▸	AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None
	▸	AlexaForBusinessReadOnlyAccess	AWS managed	None
	▸	AmazonAPIGatewayAdministrator	AWS managed	None
	▸	AmazonAPIGatewayInvokeFullAccess	AWS managed	None

9. Select "Attach existing policies directly" and provide "AdministratorAccess". This is not best practice; please go back over IAM permissions to learn about the "principle of least privileges". This is only as an example.

10. If you are happy with that, proceed to the next page. This will ask about providing
tags as key-value pairs. This is optional, so you can proceed to reviewing your
configuration, and click on "Create user" button.

11. You will be shown the access and secret key. Once you leave this page, you will no
longer be able to see your secret (this is for security purposes). AWS does allow you to
download a .csv file with these credentials. Download the csv file.


Download .csv

		User	Access key ID	Secret access key
▸	✓	sample_user	AKIA2WKJMHEZS7X4VABN	******** Show

Installing AWS CLI on Linux

This part of the guide will show you the best way to install the AWS CLI for Linux. The
machine used for this example is Ubuntu 18.04LTS.

1. Run the following commands:

```
sudo apt update -y && sudo apt install curl unzip -y
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

2. If the previous commands ran successfully, then configure the awscli to run commands as
a known user:

```
aws configure
```

3. You will be prompted to enter the "access key", "secret access key", "region" and
"output".

Both the Access and Secret Access key can be found from the .CSV file downloaded, simply
copy and paste them in.

The region should be set to "eu-west-1".

The output should be set to "json".

Installing AWS CLI on Windows

This part of the guide will show you the best way to install the AWS CLI for Windows. The machine used for this example is running Windows 10 Pro 64-bit.

1. You will need to download the installer by clicking the following link:

https://awscli.amazonaws.com/AWSCLIV2.msi

2. Run the downloaded installer and follow the instructions displayed. Usually it will ask to accept the T&C's, location for the installation and if you want to proceed with the download.

3. After installing, open up *command prompt*, and run the following to check it worked:

```
aws --version
```

4. You will need to now configure AWS so it knows who you are, run the following command:

```
aws configure
```

You will be prompted to enter the "access key", "secret access key", "region" and "output".

Both the Access and Secret Access key can be found from the .CSV file downloaded, simply
copy and paste them in.

The region should be set to "eu-west-1".

The output should be set to "json".

Resources

Overview

These are the resources used for this topic:

- IAM permissions
- IAM Users

Exercises

There are no exercises for this module.