AWS CLI

Created	@Nov 7, 2020 2:19 PM
Tags	AWS Training Work

How-to set up a basic configuration for aws-cli to ease basic operations for training courses

> Have a decent shell is a MUST 🤣
> **Linux, MacOS** → Bash, ZSH
> **Windows** → WLS → Bash, ZSH

Install AWSCLI

Installing, updating, and uninstalling the AWS CLI version 2
Install the AWS Command Line Interface version 2 (AWS CLI version 2) on your system.
https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html

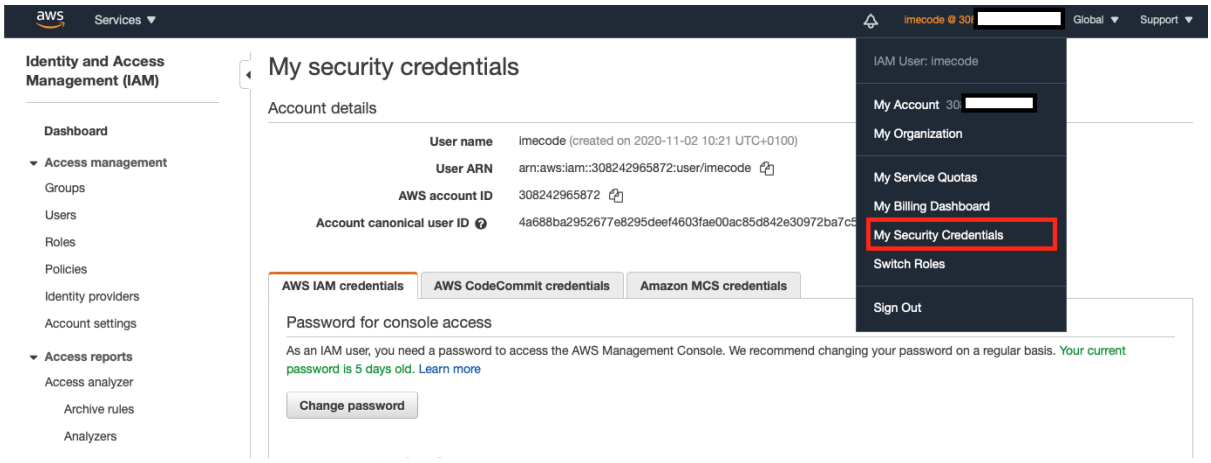
Rule of thumb:

- If you can install local software, install it locally
- If you can't and have docker, go for docker

```
# Trick for docker installations
# Make an alias for the docker run command
alias awsdocker=`docker run --rm -it -v ~/.aws:/root/.aws amazon/aws-cli`
```

Configure AWSCLI

Log in the AWS console and go to "My Security Credentials" section and create a Acces Key. You can download the configuration as a CSV file (this file should be stored securely).



AWSCLI works with profiles to use different configurations, to avoid clashing with other configurations this manual will create a profile called `sre`

```
$ aws configure --profile sre
AWS Access Key ID [None]: # Access Key ID
AWS Secret Access Key [None] # Access Secret Key
Default region name [None]: eu-west-3
Default output format [None]: json
```

For this course, we use Paris Region: *eu-west-3* because the images used in this course are located in this region.

Handling Key Pairs

We need a Key-Pair to access any EC2 instance. **To create one key pair**:

```
# Note, this output should be saved. Otherwise private key will be lost
$ aws ec2 create-key-pair --profile sre --key-name sre-kp
{
    "KeyFingerprint": "62:65:...:6a:9f:fc",
    "KeyMaterial": "[PRIVATE KEY PEM CONTENT]",
    "KeyName": "sre-kp",
    "KeyPairId": "key-00e5xxxxxxxxxxxx"
}

# To create directly a PEM file
$ aws ec2 create-key-pair --profile sre \
  --key-name sre-kp \
  --query "KeyMaterial" \
  --output text > sre-kp.pem

# Even a KeyPair has no cost it can be deleted:
$ aws ec2 delete-key-pair --profile sre --key-name sre-kp
```

AMI - Images

Before launching instances, we need to **find the image to launch**. To find by name:

```
$ aws ec2 describe-images --profile sre \
  --filters "Name=name,Values=pinchito*" \
  --query "Images[*].[Name,ImageId]"
[
    [
        "pinchito-loadtest-2020-11-07",
        "ami-0b1caa1a26cd41f9d"
    ],
    [
        "pinchito-loadtest-2020-11-06",
        "ami-0f845d0a891816ce6"
    ]
]
```

Launching EC2 instances

To launch an ec2 instance, knowing beforehand the instance id:

```
# Instance types: t2.small t2.micro t3.micro t3.small ...
# Keep the InstaceId to delete the instance after use :)
$ aws ec2 run-instances --profile sre \
  --image-id ami-0b1caa1a26cd41f9d \
  --count 1 \
  --instance-type t3.micro \
  --key-name sre-kp \
  --query "Instances[*].[InstanceId]"
[
    [
        "i-029f38af0b66790b4"
    ]
]
```

To delete an instance:

```
$ aws ec2 terminate-instances --profile sre \
  --instance-ids i-029f38af0b66790b4
```

We don't want to leave any resource running. **To check if we have any instance running:**

```
# Watch out any non terminated instance
$ aws ec2 describe-instances --profile sre \
  --query "Reservations[].Instances[].[InstanceId,State.Name]"
[
    [
        "i-029f38af0b66790b4",
        "terminated"
    ]
    [
        "i-00ba20a1ecc0fd606",
        "terminated"
    ]
]
```

ElastiCache

To **create a Redis ElastiCache** service with no read replica (Good for test and dev):

```
$ aws elasticache create-cache-cluster --profile sre \
  --cache-cluster-id sre-cache-cluser \
  --cache-node-type cache.t3.micro \
  --engine redis \
  --engine-version 3.2.4 \
  --num-cache-nodes 1 \
  --cache-parameter-group default.redis3.2
```

To get the public endpoint:

```
$ aws elasticache describe-cache-clusters --profile sre \
  --cache-cluster-id sre-cache-cluser \
  --show-cache-node-info \
  --query "CacheClusters[].[CacheClusterId,CacheNodes[].Endpoint]"
```

This is a paid service, ensure deletion to avoid over costs. To check if there is any ElastiCache running:

```
# List all ElastiCache services
$ aws elasticache describe-cache-clusters --profile sre \
  --query "CacheClusters[].[CacheClusterId,CacheClusterStatus]"

# Check an existing ElastiCache service status and info
$ aws elasticache describe-cache-clusters --profile sre \
  --cache-cluster-id sre-cache-cluser \
  --query "CacheClusters[].[CacheClusterId,CacheClusterStatus]"
```

Delete an ElastiCache

```
$ aws elasticache delete-cache-cluster --profile sre \
  --cache-cluster-id sre-cache-cluser
```

AWS Lambda

Role

To work with lambdas first we need to create a Role:

1. Define a Trust Policy
2. Create a Role for that Trust Policy (Assume Role)
3. Attach the Role with a Lambda Execution Policy

Create a file `trust-policy.json` with the Trust Policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Create the Role:

```
$ aws iam create-role --profile sre \
  --role-name sre-lambda-role2 \
  --assume-role-policy-document file://trust-policy.json
{
  ...
  "Arn": "arn:aws:iam::308242965872:role/sre-lambda-role"
  ...
}
```

Attatch Policy to Role

```
$ aws iam attach-role-policy --profile sre \
  --role-name sre-lambda-role \
  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
```

If you don`t need it anymore, **delete the role**, just in case:

```
$ aws iam detach-role-policy --profile sre \
  --role-name sre-lambda-role \
  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole

$ aws iam delete-role --profile sre --role-name sre-lambda-role
```

Function

To create a function we need to create a **Deployment Package**, a zip file containing the lambda function. The simplest node lambda function is a single `index.js` file with no dependencies:

```
zip function.zip index.js
```

Create Lambda function for nodejs12:

```
$ aws lambda create-function --profile sre \
  --function-name sre-lambda-function \
  --zip-file fileb://function.zip \
  --handler index.handler \
  --runtime nodejs12.x \
  --role arn:aws:iam::308242965872:role/sre-lambda-role
```

Invoke a Lambda

```
# The LogResult is the base64 encoded output
$ aws lambda invoke --profile sre \
  --function-name sre-lambda-function out \
  --log-type Tail
{
    "StatusCode": 200,
    "LogResult": "U1RBUlQgUmVxdWVzdElkOiA4N2QwNDRiOC1mMTU...",
    "ExecutedVersion": "$LATEST"
}
```

```
# To directly decode an output log
$ $ aws lambda invoke --profile sre \
  --function-name sre-lambda-function out \
  --log-type Tail \
  --query 'LogResult' --output text |  base64 -d

START RequestId: 57f231fb...Version: $LATEST
...
REPORT RequestId: 57f231fb...  Duration: 79.67 ms      Billed Duration: 100 ms         Memory Size: 128 MB     Max Memory Used:
```

List Lambdas

```
$ aws lambda --profile sre \
  list-functions --max-items 10
```

Get Lambda Definition

```
$ aws lambda get-function --profile sre \
  --function-name sre-lambda-function
```

Delete Lambda

```
$ aws lambda delete-function --profile sre \
  --function-name sre-lambda-function
```