Creating Identity and Access Management (IAM) Account and Policy

Objective:

- Create an IAM user account with access keys and attach a policy to it
- Make programmatic calls to AWS using access keys

General Instructions:

- Each lab is to be done individually. However you can discuss with others, but effort should finally be yours.

References:

- https://aws.amazon.com/iam/
- https://start.jcolemorrison.com/aws-ia
- m-policies-in-a-nutshell/
- https://docs.google.com/document/d/1pY9GTuMHm9uRbv5PbUcBu4mWZPrBsE-0tPo7-KBnzGM/edit?usp=sharing
- https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html
- https://docs.aws.amazon.com/cli/latest/reference/ (commands)
- https://medium.com/@luiscelismx/starting-with-aws-boto3-6a5e8c70a1ca (Boto3)
- https://docs.google.com/document/d/1WN2Hu8t3HqjeXyjODVx49c-afNru8fuZ3_4vzzj2Bx0/edit?usp=sharing

Lab Instructions:

Normally the administrator adds the users and groups, then authorizes the users for the resources and so on. In AWS, the IAM service does this for you. Root user is only Administrator at the launch of your AWS account.

An IAM account can be used to interact with AWS programmatically. One can associate permission policies to it to control what this account can and cannot do. The account is associated with access keys (a combination of access key id and a secret key) which is needed to make programmatic calls to AWS.

In this section, you will create an AWS IAM user and attach a policy you just created to the user. You will then generate the access keys for the user. Those access keys will be used to make programmatic calls to AWS services via AWS CLI or APIs.

You will first configure AWS CLI in your vlab terminal to explore the AWS CLI commands. Then

you use Boto 3(already installed in vlab) and try out some Python scripting on the terminal. Boto 3 is the AWS SDK for Python, making it easier to integrate your Python application, library, or script with AWS services.

Goal of the Activity

Suppose You are a junior cloud engineer tasked with creating an IAM policy for an Auditor role. The policy must allow certain operations the following resources across all regions for visibility and compliance checks:

Requirements

S3 Permissions:

Allow:

List all S3 buckets in the account.

Create new S3 buckets.

Download objects from any bucket.

Delete individual objects (files) in any bucket.

Block:

Deleting entire S3 buckets.

Uploading new objects to any bucket.

Lambda Permissions:

Allow:

Create new Lambda functions.

Run existing Lambda functions.

Block:

Deleting Lambda functions.

Modifying the code of existing Lambda functions.

Load Balancer Permissions:

Allow:

Create new load balancers.

Delete existing load balancers.

Block:

Viewing details about existing load balancers.

CloudWatch Permissions:

Allow:

Create new log groups.

View log entries in existing log groups.

Block:

Deleting log groups.

Adding new log entries to existing log groups.

EC2 Permissions:

Allow:

View details about EC2 instances (e.g., status, type).

View details about security groups.

View details about VPCs.

Block:

Starting and stopping EC2 instances.

IAM Permissions:

Do not modify the predefined block granting full IAM access.

Critical Rule:

The policy must not grant write/delete permissions for any service.

A predefined block allows full IAM access (iam:*) for policy verification. Do not modify or remove this block!

Task
Edit the provided **sample_iam_policy.json** template in labDirectory to achieve the requirements. Replace the comments with the correct permissions for each service.

Note: Go through the policy. Refer to https://start.jcolemorrison.com/aws-iam-policies-in-a-nutshell/ for how to interpret the policy. "*" is a wildcard making it easy to select multiple matching items associated with an action or resource.

1. Create an AWS IAM policy.

- In the AWS Management Console, click **All Services**, then under **Security, Identity and Compliance**, click **IAM** to open the **IAM dashboard**.
- In the left navigation menu, click **Policies**. Then Click **Create policy** on top right.
- There are two ways you can create IAM policies: Visual editor and JSON editor. We will use the JSON which is easier and can give fine-grained customized control over the resources. So, click the **JSON** tab.
- We will use the policy present in **sample_iam_policy.json**:
    - In the editor textbox, replace the policy with the above.
    - You may notice a security warning about the use of wildcards that are overly permissive. Normally you should follow the principle of least privilege. But for this exercise and others, this is ok.
    - Skip **Tags**.
    - Under **Review Policy**, for **Name**, type **iam-generic-policy**(or any other username you want). Skim through the summary (some have full access, some have limited access). Click **Create policy**.

2. Create an IAM account, attach policy to the account, and generate access keys

- Go back to the **IAM dashboard** if not already there.
- In the left navigation menu, click **Users**.
- Click **Add user** on top right. In the **User name** text box, type **IAMTestUser** (or any other username you want)
- For **Access type**, select both **Programmatic access and AWS Management Console access**.
- For **Console password**, you can choose a custom password and also unchecked “user must create new password at next sign-in”. But you can also choose **Autogenerated password** as such. Make sure you take a note of the password created.
- Click **Next: Permissions**. Under **Set permissions**, click **Attach existing policies directly**. Remember we already created a policy.
- In the search text box for **Filter**, type **iam-generic-policy** or whatever name you gave your policy. Select **it** from the filtered list.
- Skip **Tags.**
- Review the information and click **Create user**. You should see a success message.
    - Click **Download .csv** to download the access key ID and secret access key. **Note**: This is your only chance to download these credentials.
    - In the **Email login instructions** column, click **Send email**. You can send an email to an email address of your choice. This email contains the instructions to sign in to your AWS account .
    - Click **Close** to return to the console.

3. Configure AWS Command Line Interface (CLI) with access keys of account in vlab terminal

The AWS Command Line Interface (CLI) helps you control and configure AWS services from a terminal (you have so far been doing many of these via the browser console). You can also automate these services through scripts run on the vlab terminal. Follow the below steps to configure the CLI and access AWS services.

- All incoming requests for AWS services need to be cryptographically signed. The AWS CLI does this automatically for you. But one has to configure CLI first for this. “aws configure” command is the fastest way to set this up. When you enter this command, the AWS CLI prompts you for four pieces of information: Access key ID, Secret access key AWS Region, Output format
    - Open the **Downloads.csv** file that you downloaded earlier. There will be an entry for **IAMTestUser**. Note the values for **Access Key Id** and **Secret Access Key**.
    - On the instance terminal, type the below command.
        - `aws configure` Follow the prompts on the screen and paste in the values for **Access Key Id** and **Secret Access Key.**
        - For **Region**, type **us-west-2**.
        - For **Default output format**, press ENTER. (json is used by default)
    - You have now configured the AWS CLI so that any CLI calls will operate with the credentials of the AWS IAM user IAMTestUser.
- A detailed list of all services and the commands within are available at https://docs.aws.amazon.com/cli/latest/reference/. A few examples to try out are as under:
    - `aws ec2 describe-instances`
      You should see a JSON output with all the information of the Amazon EC2 instances in your account.
    - The above command gives all instances details. If you want a specific instance detail (e.g. the one names say IAM-EC2), you can use

      aws ec2 describe-instances --filter Name=tag:Name,Values=IAM-EC2

    - aws ec2 describe-security-groups
    - aws ec2 describe-vpcs
    - aws iam list-users (you will see all users)

4. Use of scripts to access AWS services

Boto is a Software Development kit (SDK) that helps Python developers to create, configure, and manage AWS services, such as EC2 and S3. The kit provides both

object-oriented APIs, as well as low-level APIs to access AWS services. We will see how to use this kit.

- To start using Boto 3, type **python3** on the vlab terminal and press ENTER. You should now type Python commands from your instance terminal. For example, to explore the EC2 API, type the following:

  ```
  import boto3
  client = boto3.client('ec2')
  client.describe_instances()
  ```

  You should see a JSON output similar to the one given by the AWS CLI command.
- Experiment with many other EC2 based commands based on the above structure (e.g describe-vpcs). You can Press **Ctrl-D** to exit the python interpreter.

5. Submission

- Refer to data.json, Replace **INSTRUCTOR Access key ID** and **INSTRUCTOR Secret Access Key** with IAM user credentials. Replace **region-name** with your current region and also replace **your-policy-arn** with the ARN of the policy you created for this activity.