Demo Project: Interacting with AWS CLI

This guide demonstrates how to install, configure, and utilize AWS CLI for managing AWS resources. It covers setting up AWS CLI, creating and managing EC2 instances, and interacting with IAM resources.

Step 1: Install and configure AWS CLI tool to connect to our AWS account
Step 2: Create EC2 Instance using the AWS CLI with all necessary configurations like Security Group
Step 3: Create SSH key pair
Step 4: Create EC2 instance
Step 5: Create IAM resources like User, Group, Policy using the AWS CLI
Step 6: List and browse AWS resources using the AWS CLI

Step 1: Install and configure AWS CLI tool to connect to our AWS account

1. **Install AWS CLI Version 2:**

   Follow the official guide for your operating system:

   https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

2. **Verify Installation:**

   Run the following command to confirm installation: `aws --version`

3. **Configure AWS CLI:**

   Use your **AWS Access Key ID** and **Secret Access Key** to configure AWS CLI:
   `aws configure`

   - Enter the following details:
     - AWS Access Key ID
     - AWS Secret Access Key
     - Default Region Name (e.g., `us-east-1`)
     - Default Output Format (e.g., `json`)
   - **Purpose**: This step allows AWS CLI to connect to your account and region and format its output.

4. **Verify Configuration:**

AWS CLI configurations are stored under the `~/.aws` directory. Check the files:

- `ls ~/.aws`
- `cat ~/.aws/config`
- `cat ~/.aws/credentials`

Step 2: Create EC2 Instance using the AWS CLI with all necessary configurations like Security Group

1. **List existing security groups:** `aws ec2 describe-security-groups`
2. **Identify the VPC:** `aws ec2 describe-vpcs`
3. **Create a new security group:**

   `aws ec2 create-security-group --group-name <SG name> --description <"your description"> --vpc-id <vpc id>`

   Example: `aws ec2 create-security-group --group-name my-sg --description "My SG" --vpc-id vpc-0827a287ab0e9d35c`
4. **Take note of the security group id**
5. **Add an ingress rule to allow SSH access:**

   ```
   aws ec2 authorize-security-group-ingress \
   --group-id <SG group ID> \
   --protocol tcp \
   --port 22 \
   --cidr <your IP>
   ```

   Example:

   ```
   aws ec2 authorize-security-group-ingress \
   --group-id sg-08a0d97e718f43aaa \
   --protocol tcp \
   --port 22 \
   --cidr 186.54.2.57/32
   ```
6. **Verify the Security Group Configuration:**

   `aws ec2 describe-security-groups --group-ids <SG group ID>`

Step 3: Create SSH key pair

1. **Generate an SSH key pair and save it to a file:**

   ```
   aws ec2 create-key-pair \
   --key-name <name of your key-pair> \
   --query 'KeyMaterial' \
   --output text > <file_name>.pem
   ```

   Example:

   ```
   aws ec2 create-key-pair \
   --key-name MyKpCli \
   --query 'KeyMaterial' \
   --output text > MyKpCli.pem
   ```

2. **Set Proper File Permissions:** `chmod 400 MyKpCli.pem`

   - **Why is this necessary?**

     SSH requires that the private key file be readable **only by the owner**. If the file is accessible to other users (e.g., group or world permissions), SSH will refuse to use it, resulting in the **"bad permissions"** or **"Permission denied"** error.

     Without `chmod 400`, attempting to SSH into the EC2 instance will fail with an error like:

     ```
     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     @         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     Permissions 0644 for 'MyKpCli.pem' are too open.
     It is required that your private key files are NOT accessible by
     others.
     This private key will be ignored.
     Load key "MyKpCli.pem": bad permissions
     Permission denied (publickey).
     ```

3. **Verify the Key Pair:** `ls -l MyKpCli.pem`

Step 4: Create EC2 instance

1. **Identify and copy the subnet:** `aws ec2 describe-subnets`
2. **Go to the AWS console, choose and copy the AMI for the instance.**
3. **Launch the EC2 Instance:**

```
aws ec2 run-instances \
--image-id <ami-id> \
--count 1 \
--instance-type t2.micro \
--key-name MyKeyPair \
--security-group-ids <sg-id> \
--subnet-id <subnet-id>
```

Example:

```
aws ec2 run-instances \
--image-id ami-01816d07b1128cd2d \
--count 1 \
--instance-type t2.micro \
--key-name MyKpCli \
--security-group-ids sg-08a0d97e718f43aaa \
--subnet-id subnet-063497bab3bbcab10
```

5. **Retrieve the public IP address:** `aws ec2 describe-instance`
6. **SSH into the Instance:** `ssh -i MyKeyPair.pem ec2-user@<public-ip>`

   Example: `ssh -i MyKpCli.pem ec2-user@3.89.218.122`

Step 5: Create IAM resources like User, Group, Policy using the AWS CLI

1. **Create a user group:** `aws iam create-group --group-name MyGroupCli`
2. **Create the user:** `aws iam create-user --user-name MyUserCli`
3. **Add the user to the group:**

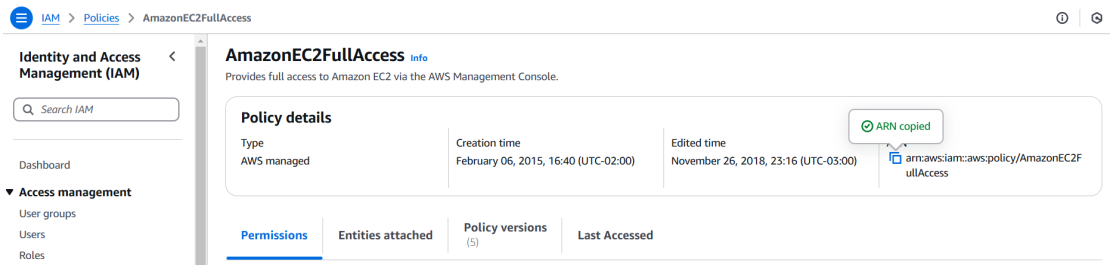
   `aws iam add-user-to-group --user-name MyUserCli --group-name MyGroupCli`
4. **Check the group information to confirm the user was added:**

   `aws iam get-group --group-name MyGroupCli`

5. **Attach a Managed Policy to the Group:**

`aws iam attach-group-policy --group-name MyGroupCli --policy-arn arn:aws:iam::aws:policy/AmazonEC2FullAccess`

- In case you need to find the Managed Policy you can obtain via...
  - AWS Console → IAM → Policies → look for “AmazonEC2Full Access” → copy the ARN



  - or... in the AWS CLI:

    `aws iam list-policies --query 'Policies[?PolicyName==`AmazonEC2FullAccess`].Arn' --output text`

    Output example:

```
$ aws iam list-policies --query 'Policies[?PolicyName==`AmazonEC2FullAccess`].Arn' --output text
arn:aws:iam::aws:policy/AmazonEC2FullAccess
```

6. **Verify Attached Policies:**

`aws iam list-attached-group-policies --group-name MyGroupCli`

7. **Create a Login Profile for Console Access:**

`aws iam create-login-profile --user-name MyUserCli --password MyPassword! --password-reset-required`

Note: When User Login into the AWS console, a prompt will require the user to change the password.

8. **Create a Custom Policy for Password Changes:**

- Copy this json file:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:GetAccountPasswordPolicy",
      "Resource": "*"
    },
```

```
    {
      "Effect": "Allow",
      "Action": "iam:ChangePassword",
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ]
}
```

Reference: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_passwords_enable-user-change.html

- Create a JSON file and and paste the json here: `vim changePwdPolicy.json`
- Create the policy: `aws iam create-policy --policy-name changePwd --policy-document file://changePwdPolicy.json`

9. **Attach the Custom Policy to the Group:**

   `aws iam attach-group-policy --group-name MyGroupCli --policy-arn arn:aws:iam::038462748802:policy/changePwd`

10. **Create an Access Key for the User:**

    `aws iam create-access-key --user-name MyUserCli`

11. **Switch AWS CLI credentials temporarily:**

    - `export AWS_ACCESS_KEY_ID=<access-key-id>`
    - `export AWS_SECRET_ACCESS_KEY=<secret-access-key>`

12. **Test the User's Permissions:**

    - List EC2 instances: `aws ec2 describe-instances`
    - Attempt restricted actions: `aws iam create-user --user-name TestUser`

      You should get an error like this:

```
An error occurred (AccessDenied) when calling the CreateUser operation:
User: arn:aws:iam::038462748802:user/MyUserCli is not authorized to per
form: iam:CreateUser on resource: arn:aws:iam::038462748802:user/test b
ecause no identity-based policy allows the iam:CreateUser action
```

Step 6: List and browse AWS resources using the AWS CLI

1. **List AWS Resources:**
   - Describe EC2 instances: `aws ec2 describe-instances`
2. **Delete AWS Resources:**
   You can see the list by
   `aws iam aa` , look through all the “delete-.....” commands to decide which to delete:

```
delete-access-key                       | delete-account-alias
delete-account-password-policy          | delete-group
delete-group-policy                     | delete-instance-profile
delete-login-profile                    | delete-open-id-connect-provider
delete-policy                           | delete-policy-version
delete-role                             | delete-role-permissions-boundary
delete-role-policy                      | delete-saml-provider
delete-ssh-public-key                   | delete-server-certificate
delete-service-linked-role              | delete-service-specific-credential
delete-signing-certificate              | delete-user
delete-user-permissions-boundary        | delete-user-policy
```