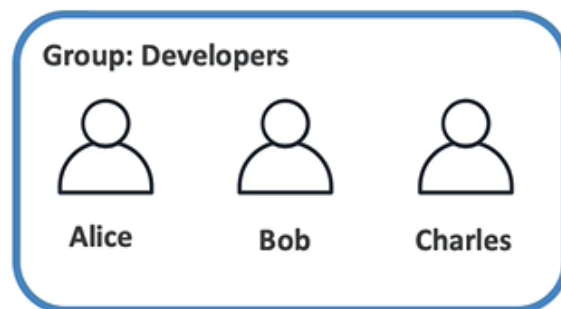


IAM: Users & Groups

1. IAM stands for Identity and Access Management and it is a global service..in IAM we r going to create users and assign them groups
2. Root Account should not be used or shared..
3. What you should be doing instead, is create users.
4. [So you will create users in IAM.](#)

IAM: Users & Groups

- IAM = Identity and Access Management, Global service
 - Root account created by default, shouldn't be used or shared
 - Users are people within your organization, and can be grouped
- 5.
 6. We have created 6 people and all of them are in our organisation and we have created groups based on their roles



7.
 - Groups only contain users, not other groups
 - Users don't have to belong to a group, and user can belong to multiple groups
- 8.
9. A user can work for multiple groups ..



- 10.
11. IAM Permission: This is just describing in, I think plain English,
12. what a user is allowed to do or what a group
13. and all the users within that group are allowed to do. It comes in the JSON documents called policies

IAM: Permissions

- Users or Groups can be assigned JSON documents called policies

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "elasticloadbalancing:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:Describe*"
      ],
      "Resource": "*"
    }
  ]
}
```

Now we'll see what EC2
elastic load balancing

- 14.
15. Here in the picture we can see ...that we have allowed users to use ec2,elasticloadbalancing and cloudwatch
16. It help us to give permissions to users
17. We have a principle called least privilege principle

IAM: Permissions

- Users or Groups can be assigned JSON documents called policies
 - These policies define the permissions of the users
 - In AWS you apply the least privilege principle: don't give more permissions than a user needs
- 18.
19. Now if a user wants 3 services ... We just create permission for that user

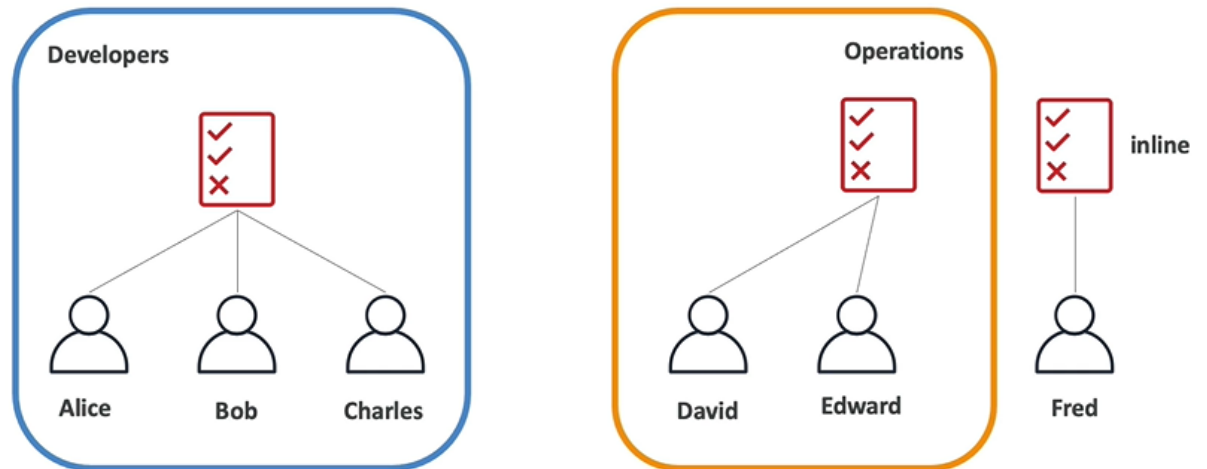
IAM hands on:

1. As the root user have many permission..we create admin users and regular users
2. To create an user we can go to IAM -> users
3. After that we have create a group called admin and gave administrative policy permission
4. Every permission we assign to group..it can be assigned to user
5. After that we have created Account alias..as account number will be hard to remember
6. After that we will login as IAM user

IAM Policies:

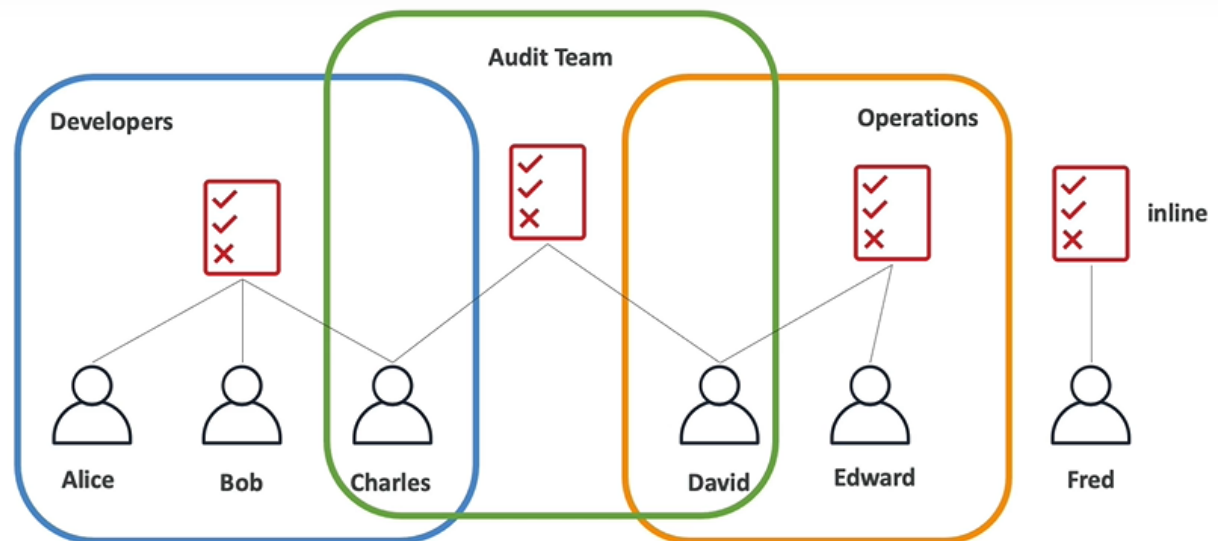
1. Every group has their own set of policies...like developer group can have some policies which are different from Operations group
2. If Fred is a user, it has the possibility not to belong to a group. And we have the possibility to create what's called an inline policy which has a policy [that's only attached to a user.](#)

IAM Policies inheritance



- 3.
4. And if Charles and David are in multiple group..like for example here they both are in audit group..now they have their own set of policies

IAM Policies inheritance



- 5.

```
{
  "Version": "2012-10-17",
  "Id": "S3-Account-Permissions",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
        "AWS": ["arn:aws:iam::123456789012:root"]
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::mybucket/*"]
    }
  ]
}
```

6. IAM Policies Structure

7. It consists of version policy language version ..we'll always include "2012-10-17"

- Consists of
 - Version: policy language version, always include "2012-10-17"
 - Id: an identifier for the policy (optional)
 - Statement: one or more individual statements (required)

• Statements consists of

- Sid: an identifier for the statement (optional)
- Effect: whether the statement allows or denies access (Allow, Deny)
- Principal: account/user/role to which this policy applied to
- Action: list of actions this policy allows or denies
- Resource: list of resources to which the actions applied to
- Condition: conditions for when this policy is in effect (optional)

```
{
  "Version": "2012-10-17",
  "Id": "S3-Account-Permissions",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
        "AWS": ["arn:aws:iam::123456789012:root"]
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::mybucket/*"]
    }
  ]
}
```

8. .

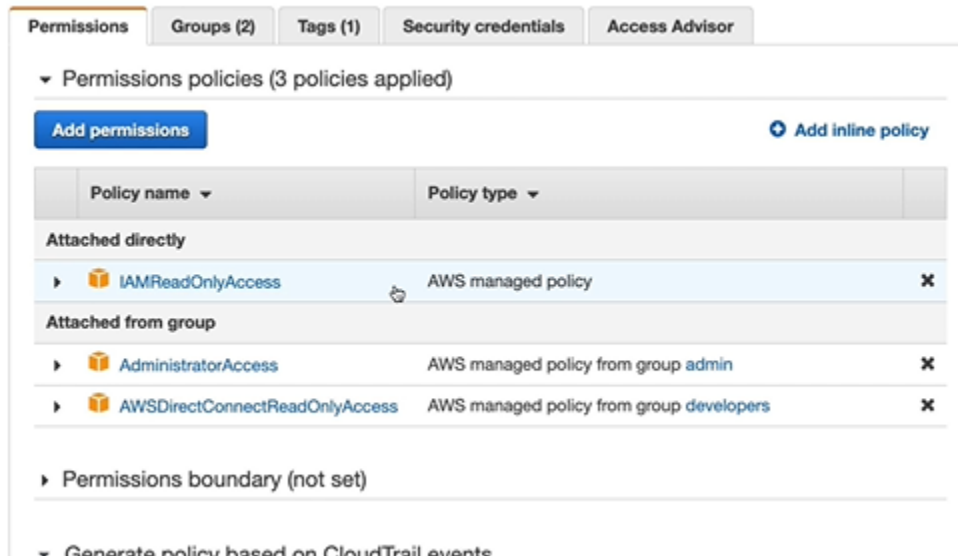
9. Here in statement we have effects...it allows or denies access to certain API's

10. In principle here we are applying our policies to the root user

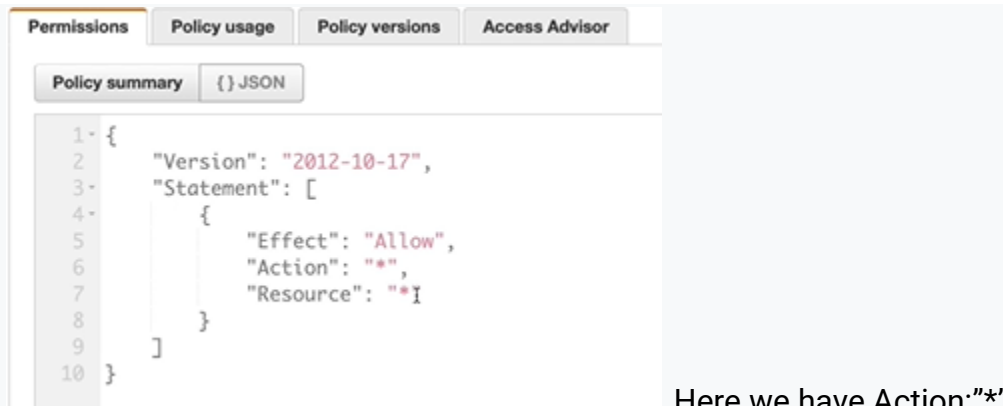
IAM Hands On:

1. Here like if a user is removed from the group..The policies of the user will also be removed

2. We can also direct attach some policies to the user ..we have attached IAMReadOnlyAccess to the user directly
3. Later we have added him again to admin group as the admin group has Administrative Access policy user will also get that
4. We have added same user to the developer group and gave AWSDirectConnectReadOnlyAccess



5. We can see the policies got inherited from different ways through the IAM permissions
6. If we check policies tab...here we have all the policies of IAM
7. Here we can see administrativeAccess policy...



9. Here we have Action:"*", Resources:"*" .. So we allow all the actions from all the resources

10. We have a total of

The screenshot shows the AWS IAM console interface for a policy. At the top, there are four tabs: 'Permissions' (highlighted in orange), 'Entities attached', 'Policy versions', and 'Access Advisor'. Below the tabs, the title 'Permissions defined in this policy' is followed by a blue 'Info' link. A descriptive text states: 'Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.' To the right of this text are two buttons: 'Summary' and 'JSON'. Below the text is a search bar with a magnifying glass icon and the placeholder text 'Search'. At the bottom left, it says 'Allow (379 of 379 services)'. At the bottom right, there is a toggle switch labeled 'Show remaining 0 services'.

services to avail

11. We can also create our own policies

IAM MFA Overview:

1. We can define password policy to protect our users and groups from being compromised
2. We can give some set of rules for password and Allow Users to change their passwords
3. Or We can say change your password for every 90days
4. Also we can prevent re-use of same password

IAM – Password Policy

- Strong passwords = higher security for your account
- In AWS, you can setup a password policy:
 - Set a minimum password length
 - Require specific character types:
 - including uppercase letters
 - lowercase letters
 - numbers
 - non-alphanumeric characters
 - Allow all IAM users to change their own passwords
 - Require users to change their password after some time (password expiration)

5.

6. Multi Factor Authentication:

Multi Factor Authentication - MFA

- Users have access to your account and can possibly change configurations or delete resources in your AWS account
- You want to protect your Root Accounts and IAM users
- MFA = password *you know* + security device *you own*

7.



Alice

Password

+



=>

Successful login

- Main benefit of MFA:
if a password is stolen or hacked, the account is not compromised

8.

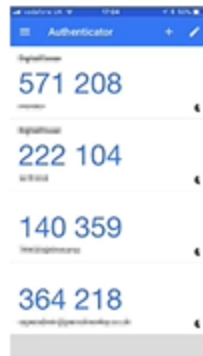
9. To enable MFA we have diff options in AWS

10. We can use google authenticator in mobiles or by using authy(it works on both phones and lappy)

11. In authy we can get multiple tokens(for root user,IAM user etc) on a single device

MFA devices options in AWS

Virtual MFA device



Google Authenticator
(phone only)



Authy
(multi-device)

Support for multiple tokens on a single device.

12.

Universal 2nd Factor (U2F) Security Key



YubiKey by Yubico (3rd party)

Support for multiple root and IAM users
using a single security key

13.

MFA devices options in AWS

Hardware Key Fob MFA Device



Provided by Gemalto (3rd party)

Hardware Key Fob MFA Device for AWS GovCloud (US)



Provided by SurePassID (3rd party)

14.

IAM MFA Hands On:

1. Just see online..it is just adding 2 factor authentication

AWS Access Keys, CLI and SDK

1. We have 3 ways to Access AWS

How can users access AWS ?



- To access AWS, you have three options:
 - AWS Management Console (protected by password + MFA)
 - AWS Command Line Interface (CLI): protected by access keys
 - AWS Software Developer Kit (SDK) - for code: protected by access keys

2.



3. AWS CLI is a command line interface which is protected by access keys and access key are credentials which will allow us to access AWS from our terminal
4. AWS SDK which is used to call API's from AWS from within our application code.

- Access Keys are generated through the AWS Console
- Users manage their own access keys
- Access Keys are secret, just like a password. Don't share them

- 5.
6. Users should manage their own Access keys

- Access Keys are secret, just like a password. Don't share them
- Access Key ID ~= username
- Secret Access Key ~= password

7.

Example (Fake) Access Keys

Access keys

Use access keys to make secure REST or HTTP Query protocol requests to AWS service APIs. For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation. [Learn more](#)

Create access key

Access key ID	Created	Last used	Status	
AKIAASK4E37PV4TU3RD6C	2020-05-25 15:13 UTC+0100	N/A	Active	Make inactive ✕

- Access key ID: AKIAASK4E37PV4983d6C
- Secret Access Key: AZPN3z0jWozWCndljhB0Unh8239aI bzbzO5fqkZq
- Remember: don't share your access keys

- 8.
9. With this Access key id and secret access key ...these when loaded into CLI will allow me to Access the AWS API

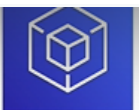
What's the AWS CLI?

- A tool that enables you to interact with AWS services using commands in your command-line shell
- Direct access to the public APIs of AWS services
- You can develop scripts to manage your resources
- It's open-source <https://github.com/aws/aws-cli>
- Alternative to using AWS Management Console

```
→ ~ aws s3 cp myfile.txt s3://ccp-mybucket/myfile.txt
upload: ./myfile.txt to s3://ccp-mybucket/myfile.txt
→ ~ aws s3 ls s3://ccp-mybucket
2021-05-14 03:22:52          0 myfile.txt
→ ~
```

10.

What's the AWS SDK?



- AWS Software Development Kit (AWS SDK)
- Language-specific APIs (set of libraries)
- Enables you to access and manage AWS services programmatically
- Embedded within your application
- Supports
 - SDKs (JavaScript, Python, PHP, .NET, Ruby, Java, Go, Node.js, C++)
 - Mobile SDKs (Android, iOS, ...)
 - IoT Device SDKs (Embedded C, Arduino, ...)
- Example: AWS CLI is built on AWS SDK for Python



Your Application

11.

12. We have to integrate AWS SDK inside our application

AWS CLI on Windows:

1. Just follow instructions

AWS CLI Hands on:

1. We can create access key ..by going to users->kaushik->security credentials->..scroll down... create access key
2. After generating aws access key for CLI..open cmd and type

```
~ aws configure
AWS Access Key ID [None]: AKIA40HDCKKLTZNZBRUT
AWS Secret Access Key [None]: D32MzVn1IqqrPPYxvWAtvx0Mlt03ft6I/Dyan6PF
Default region name [None]: eu-west-1
```

3.

```
C:\Users\iamka>aws iam list-users
{
  "Users": [
    {
      "Path": "/",
      "UserName": "Kaushik",
      "UserId": "AIDAQUSPXRYVFR53PRXFG",
      "Arn": "arn:aws:iam::044190633514:user/Kaushik",
      "CreateDate": "2023-06-21T00:59:52+00:00",
      "PasswordLastUsed": "2023-06-21T01:07:38+00:00"
    },
    {
      "Path": "/",
      "UserName": "Kaushik_winkart",
      "UserId": "AIDAQUSPXRYVDYJP7IJ7D",
      "Arn": "arn:aws:iam::044190633514:user/Kaushik_winkart",
      "CreateDate": "2023-06-11T19:43:53+00:00",
      "PasswordLastUsed": "2023-06-12T19:02:24+00:00"
    }
  ]
}
```

- 4.
5. If we type aws iam list-users we get all the information same as console
6. Even if we change anything in aws console..CLI will reflect same

AWS CloudShell:

1. Cloud shell is nothing but terminal on cloud
2. In cloudshell pwd gives full path to file in aws

```
[cloudshell-user@ip-10-0-184-1 ~]$ ls
[cloudshell-user@ip-10-0-184-1 ~]$ echo "test" > demo.txt
[cloudshell-user@ip-10-0-184-1 ~]$ cat demo.txt
test
[cloudshell-user@ip-10-0-184-1 ~]$ ls
demo.txt
[cloudshell-user@ip-10-0-184-1 ~]$ pwd
/home/cloudshell-user
[cloudshell-user@ip-10-0-184-1 ~]$
```

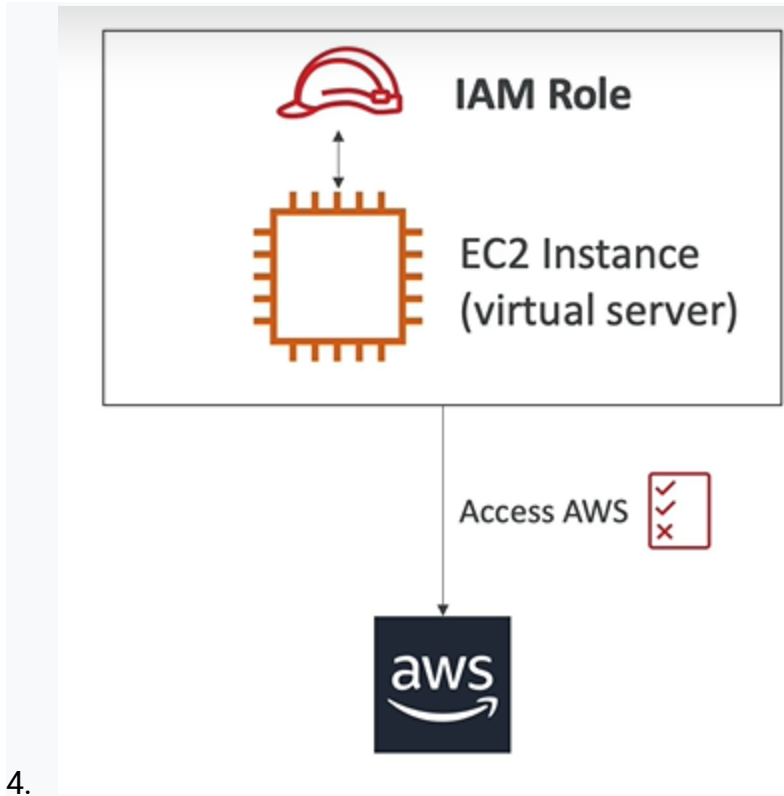
- 3.
4. We can either use cloudshell or CLI...but prefer cloudshell bcuz it has more options

IAM Roles:

IAM Roles for Services

- Some AWS service will need to perform actions on your behalf
- To do so, we will assign permissions to AWS services with IAM Roles

- 1.
2. IAM roles are similar to users...they will need some kind of permissions..so we need to assign them
3. So these IAM role will be just like a user, but they are intended to be used not by physical people, but instead they will be used by AWS services.



5. Here when we create an EC2 instance..it might need some permission to access AWS..so to avail them it uses IAM Role

- 6.
- Common roles:
 - EC2 Instance Roles
 - Lambda Function Roles
 - Roles for CloudFormation

IAM Roles Hands On:

1. To create roles..we go to roles section and click create role
2. And click on AWS servies and everything is out of syllabus ..then choose Ec2
3. Then we give policy permission and a name to the role

IAM security Tools:

1. So we can create an IAM Credentials Report
2. and this is at your account-level, This report will contain all your accounts users [and the status of their various credentials.](#)
3. We can also create IAM Access Advisor which is on User level

IAM Security Tools

- IAM Credentials Report (account-level)
 - a report that lists all your account's users and the status of their various credentials
- IAM Access Advisor (user-level)
 - Access advisor shows the service permissions granted to a user and when those services were last accessed.

4.

IAM security Hands on:

1. On bottom left we have credential report and we can download it
2. We can access "Access Advisor" by going to user profile and click on it

IAM guidelines and best practices

IAM Guidelines & Best Practices

- Don't use the root account except for AWS account setup
- One physical user = One AWS user
- Assign users to groups and assign permissions to groups
- Create a strong password policy
- Use and enforce the use of Multi Factor Authentication (MFA)
- Create and use Roles for giving permissions to AWS services

1.

- Use Access Keys for Programmatic Access (CLI / SDK)
- Audit permissions of your account using IAM Credentials Report & IAM Access Advisor

2.

- Never share IAM users & Access Keys

Shared Responsibility Model for IAM

1. Aws is responsible for everything they do like infrastructure , configuration and vulnerability analysis
2. So AWS is responsible for everything that they do, for example, their infrastructure
3. and their global network security, the configuration and vulnerability analysis of the services they offer, and any sign of compliance that they are responsible for.
4. You are responsible for users,...

Shared Responsibility Model for IAM



- Infrastructure (global network security)
- Configuration and vulnerability analysis
- Compliance validation



You

- Users, Groups, Roles, Policies management and monitoring
- Enable MFA on all accounts

5.



You

- Users, Groups, Roles, Policies management and monitoring
- Enable MFA on all accounts
- Rotate all your keys often
- Use IAM tools to apply appropriate permissions
- Analyze access patterns & review permissions

6.

7.

IAM Section Summary:

1.

IAM Section – Summary

- Users: mapped to a physical user; has a password for AWS Console
- Groups: contains users only
- Policies: JSON document that outlines permissions for users or groups
- Roles: for EC2 instances or AWS services
- Security: MFA + Password Policy
- AWS CLI: manage your AWS services using the command-line
- AWS SDK: manage your AWS services using a programming language
- Access Keys: access AWS using the CLI or SDK
- Audit: IAM Credential Reports & IAM Access Advisor