SECTION 02: IAM

USERS & GROUP:

- IAM: GLOBAL SERIVCE
- ROOT ACC CREATED BY DEFAULT
- Users within org. can be grouped
- Users can belong to 0 or multiple groups

IAM PERMISSIONS:

- Policies:

1. users or groups can be assigned JSON doc called policies.
2. defines permissions of users
3. LEAST PRIVILEGE PRINCIPLE:

> 💡 **DON’T GIVE MORE PERMISSIONS THAN A USER NEEDS.**

- Note: Inline policies are attached to 1 user only.(if not in any group)

IAM POLICIES STRUCTURE:

1. version: policy language version
2. id: an identifier for policy(optional)
3. statement: 1 or more (required)

- sid: identifier for statement (optional)
- effect: allow/deny access

- principal: which acc/user/role to apply policy (arn link)
- action: list of api/actions calls to deny or allow
- resources: resource list to which action is applied
- condition: condition for which this policy will be applied(optional)

MFA-MULTI FACTOR AUTHENTICATION:

- PROTECT YOUR ROOT ACC AND IAM USERS.
- MFA=password u know+security device u own
- Types:

1. virtual MFA: support for multiple tokens on a single device. ex: google authenticator,authy
2. universal 2nd factor(U2F) security key: supports for multiple root and iam users using single security key.ex:yubikey by yubico(3rd party)
3. hardware key fob MFA device: provided by gemalto(3rd party)
4. hardware key fob MFA device for AWS GovCloud(US): provided by SurePassID(3rd party)

AWS USER ACCESS KEYS

- Generated through AWS console
- to access aws we have 3 options:

1. AWS management console // password+MFA
2. AWS Command line interface(CLI) // access keys

   *enables you to interacrt with aws services using commands in comand-line shell

   *direct access to public APIs of AWS services

   *can develop scripts to manage aws resources

*alternative to aws management console

3. AWS Software developer kit (SDK)- for code //access keys

   *Language specific APIs

   *access aws services programmatically

   *embedded within your application

   *supports sdks(javascript,python,php,.net,ruby,java,go,node.js,c++),mobile sdk(andriod,ios),iot devices sdk(embedded,c,arduino)

   *ex: AWS CLI is built on AWS SDK for python.

IAM ROLES for Services

- some AWS services will need to perform actions on your behalf, to do so, we will assign permissions to AWS services with IAM Roles.
- common roles: ec2 instance roles,lambda function roles,roles for cloud formation

IAM SECURITY TOOLS

1. IAM Credential report(account-level): report that lists all your account's users and status of their various credentials
2. IAM Access advisor(user -level): access advisor shows service permissions granted to a user and when those services were last accessed. can be used to revise your policies.

SHARED RESPONSIBILTY MODEL FOR IAM:

- AWS:

1. Infrastructure (global network security)
2. configuration and vulnerability analysis
3. compliance validation

- YOU:

1. Users,groups,roles,policies management and monitoring
2. Enable MFA on all accounts.
3. Rotate all your keys often
4. Use IAM tools to apply appropriate permissions
5. Analyze access patterns and review permissions

SUMMARY

- Users: mapped to aphysical user, has 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 command-line.
- AWS SDK: manage your AWS services using programming language
- Access Keys: access AWS using CLI or SDK
- Audit: IAM Credential Reports & IAM Access Advisor