**Document Title**: Lambda Setup Document

**Project Title**: Hatching Sparrow CRM

**Date**: 6/27/2024

**Author(s)**: Ryan O’Connor, Full Stack Software Engineer

**Confidentiality:** N/A

**Comprehensive Guide to Using IAM Roles and Temporary Security Credentials in AWS**

**Introduction**

In AWS, managing access and permissions is crucial for maintaining security and efficient operations. One effective way to grant programmatic access to multiple users is by using IAM Roles with temporary security credentials. This guide will explain the key concepts, steps, and mechanisms involved in creating, managing, and using IAM Roles and temporary security credentials.

**Table of Contents**

1. Understanding Key Concepts
   * What is an ARN?
   * What is root in IAM?
2. Setting Up IAM Roles
   * Creating a Role
   * Editing the Trust Policy
3. Using Temporary Security Credentials
   * How Junior Engineers Assume the Role
   * Setting Environment Variables
4. Mechanism of Creation, Management, and Destruction
   * Creation
   * Management
   * Destruction
5. Making Resources Visible to Everyone in the Organization
6. Summary
7. Additional Resources

**1. Understanding Key Concepts**

**What is an ARN?**

* **ARN (Amazon Resource Name)**: A unique identifier for AWS resources. For example, arn:aws:iam::123456789012:role/JuniorEngineerRole.

**What is root in IAM?**

* **root**: The top-level account in AWS that has full access to all resources. In a trust policy, root means any user or role within that AWS account can assume the role if they have the necessary permissions.

**2. Setting Up IAM Roles**

**Creating a Role**

1. **Go to the AWS Management Console**.
2. **Navigate to IAM (Identity and Access Management)**.
3. **Create a New Role**:
   * Choose "Another AWS account" and enter the account IDs of the junior engineers' AWS accounts.
   * Attach policies to the role that grant necessary permissions (e.g., access to S3, Lambda, DynamoDB).

**Editing the Trust Policy**

To allow multiple accounts to assume the role, you need to add each account's root ARN to the trust policy.

**Example Trust Policy for Multiple Accounts:**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"AWS": [

"arn:aws:iam::123454656788:root",

"arn:aws:iam::123456789012:root",

"arn:aws:iam::234567890123:root",

"arn:aws:iam::345678901234:root",

"arn:aws:iam::456789012345:root"

]

},

"Action": "sts:AssumeRole"

}

]

}

1. **Navigate to the Role**: In the IAM console, select the role you created for the junior engineers.
2. **Edit Trust Relationships**: Go to the "Trust relationships" tab and edit the trust policy to include multiple accounts as shown above.

**3. Using Temporary Security Credentials**

**How Junior Engineers Assume the Role**

Junior engineers need to assume the role to get temporary security credentials.

**Steps:**

1. **Open a Terminal or Command Prompt**.
2. **Use the aws sts assume-role Command**:

aws sts assume-role --role-arn arn:aws:iam::<management-account-id>:role/<role-name> --role-session-name <session-name> --duration-seconds 3600

**Example Command:**

aws sts assume-role --role-arn arn:aws:iam::123456789012:role/JuniorEngineerRole --role-session-name JuniorEngineerSession

**Setting Environment Variables**

After obtaining the temporary credentials, set environment variables to use them with AWS CLI or SDKs.

**Example Response from sts:AssumeRole:**

{

"Credentials": {

"AccessKeyId": "ASIA....",

"SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",

"SessionToken": "IQoJb3JpZ2luX2VjEO//////////wEaCXVzLXdlc3QtMiJIMEYCIQ....",

"Expiration": "2023-12-27T20:28:08Z"

},

"AssumedRoleUser": {

"AssumedRoleId": "AROA....",

"Arn": "arn:aws:sts::123456789012:assumed-role/JuniorEngineerRole/JuniorEngineerSession"

}

}

**Set Environment Variables:**

export AWS\_ACCESS\_KEY\_ID=ASIA....

export AWS\_SECRET\_ACCESS\_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

export AWS\_SESSION\_TOKEN=IQoJb3JpZ2luX2VjEO//////////wEaCXVzLXdlc3QtMiJIMEYCIQ....

**4. Mechanism of Creation, Management, and Destruction**

**Creation**

* **Role Assumption**: User calls sts:AssumeRole with the role ARN and duration.
* **AWS STS (Security Token Service)**: AWS generates temporary credentials and returns them to the user.

**Management**

* **Environment Variables**: Temporary credentials are stored in environment variables for AWS CLI or SDKs.
* **Credential Rotation**: Users need to call sts:AssumeRole again to get new credentials before the current ones expire.

**Destruction**

* **Automatic Expiration**: Temporary credentials expire automatically after the specified duration.
* **No Manual Deletion Required**: AWS handles the expiration and invalidation of credentials.

**5. Making Resources Visible to Everyone in the Organization**

To make resources like S3 buckets, Lambda functions, and DynamoDB tables visible and accessible to everyone in the organization, you need to set up resource-based policies that grant access to the entire organization.

**S3 Buckets**

1. **Update the S3 Bucket Policy**:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:\*",

"Resource": [

"arn:aws:s3:::<bucket-name>",

"arn:aws:s3:::<bucket-name>/\*"

],

"Condition": {

"StringEquals": {

"aws:PrincipalOrgID": "o-xxxxxxxxxx"

}

}

}

]

}

**Lambda Functions**

1. **Update the Lambda Function's Resource Policy**:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": "\*",

"Action": "lambda:InvokeFunction",

"Resource": "arn:aws:lambda:<region>:<account-id>:function:<function-name>",

"Condition": {

"StringEquals": {

"aws:PrincipalOrgID": "o-xxxxxxxxxx"

}

}

}

]

}

**DynamoDB Tables**

1. **Update the DynamoDB Table's Resource Policy**:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": "\*",

"Action": "dynamodb:\*",

"Resource": [

"arn:aws:dynamodb:<region>:<account-id>:table/<table-name>",

"arn:aws:dynamodb:<region>:<account-id>:table/<table-name>/\*"

],

"Condition": {

"StringEquals": {

"aws:PrincipalOrgID": "o-xxxxxxxxxx"

}

}

}

]

}

**6. Summary**

Using IAM Roles with temporary security credentials offers a secure and efficient way to manage access for multiple users in AWS. By following the steps outlined in this guide, you can create roles, configure trust policies, and use temporary credentials to access AWS resources programmatically. Additionally, setting up resource-based policies ensures that your team can access the necessary resources without compromising security.

**7. Additional Resources**

* [AWS IAM User Guide](https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html)
* [AWS Security Token Service (STS) Documentation](https://docs.aws.amazon.com/STS/latest/APIReference/Welcome.html)
* [Best Practices for IAM](https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html)

This guide provides a comprehensive overview of using IAM Roles and temporary security credentials, ensuring your AWS environment remains secure and accessible to your team.