IAM Role:

- Role is an IAM identity used to provide temporary access to users and services or applications.
- Instead of being uniquely associated with one person, **a role is intended to be assumable by anyone** who needs it.
- A role **does not have standard long-term credentials such as a password or access keys** associated with it.
- When you assume a role, it provides you with temporary security credentials for your role session.
- Instead of attaching permissions to user or group directly we create a role and attach a permission to Role, user has to assume the role to access the permissions attached to the Role.
- When user assume/switch role he uses the permissions attached to the role and leaves his own permissions until role is assumed.
- AWS permissions are granted to a user by associating the user with a role.
- A user can be associated with multiple roles.
- Each role has one or more policies attached.
- User need STS permission attached and Account ID and Role name to assume the role.

Roles can be used for:

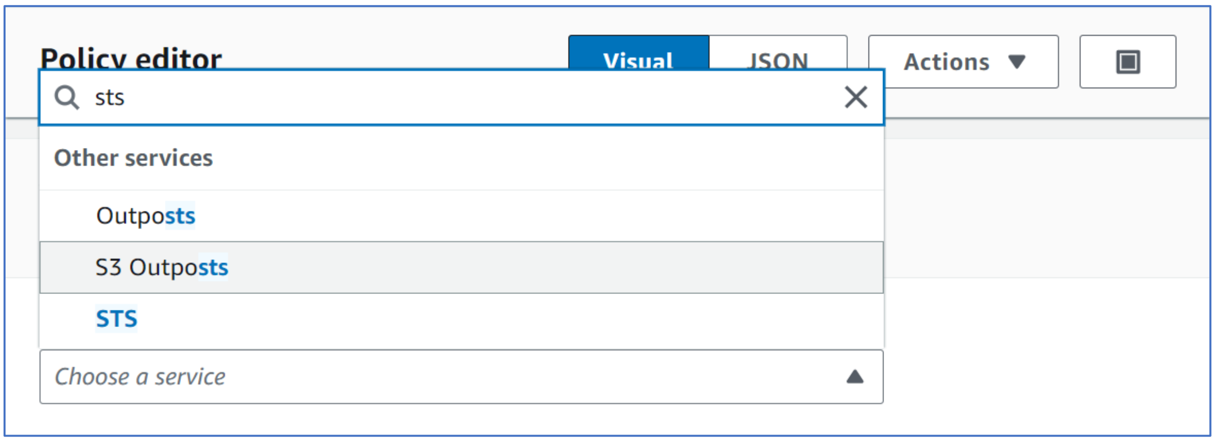
A. **Same account**
B. **Cross Account**
C. **Service to Service Access**

Steps to create Role and Attach STS permission:

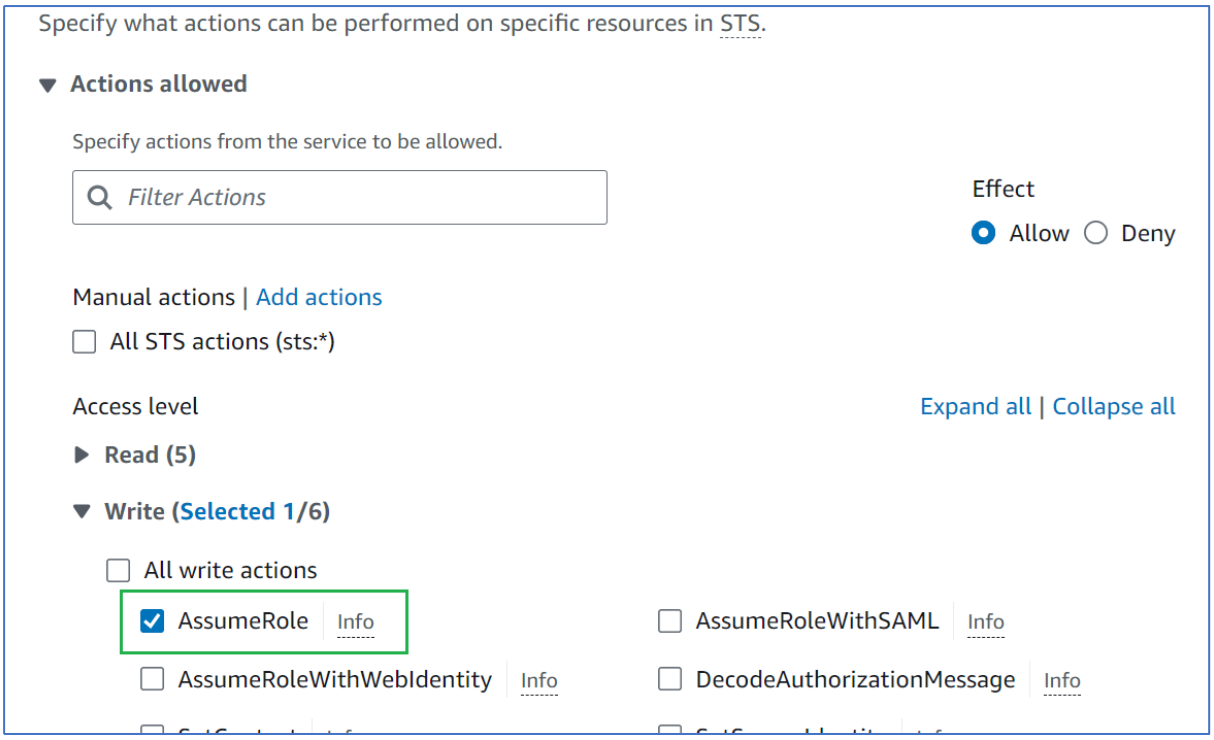
Go to IAM dashboard ---> Click on 'Roles' ---> Click on 'Create Role' ---> Click on 'AWS account' ---> Select 'This account' ---> Click on 'next' ---> Attach required policy to user (Eg: S3 Full access) ---> Click on 'Next' ---> Give Role name & Description ---> Click on 'Create Role' ---> Now login with the IAM user ---> On the top right corner click on 'user name' ---> Click on 'Switch role' ---> Enter 'Account ID', 'Role name' & Display name (Optional) ---> Click on 'Switch Role'

To assume role, you must attach - '**STS: AssumeRole** permission' to user

STEP 1: To assign STS permission to user ---> go to user ---> permissions ---> inline policy and select service STS:



STEP 2: Select Assume Role actions from the Write Category:



Specify what actions can be performed on specific resources in STS.

Actions allowed

Specify actions from the service to be allowed.

Filter Actions

Effect

[✓ Allow] Deny

Manual actions | Add actions

☐ All STS actions (sts:*)

Access level

Expand all | Collapse all

Read (5)

Write (Selected 1/6)

☐ All write actions

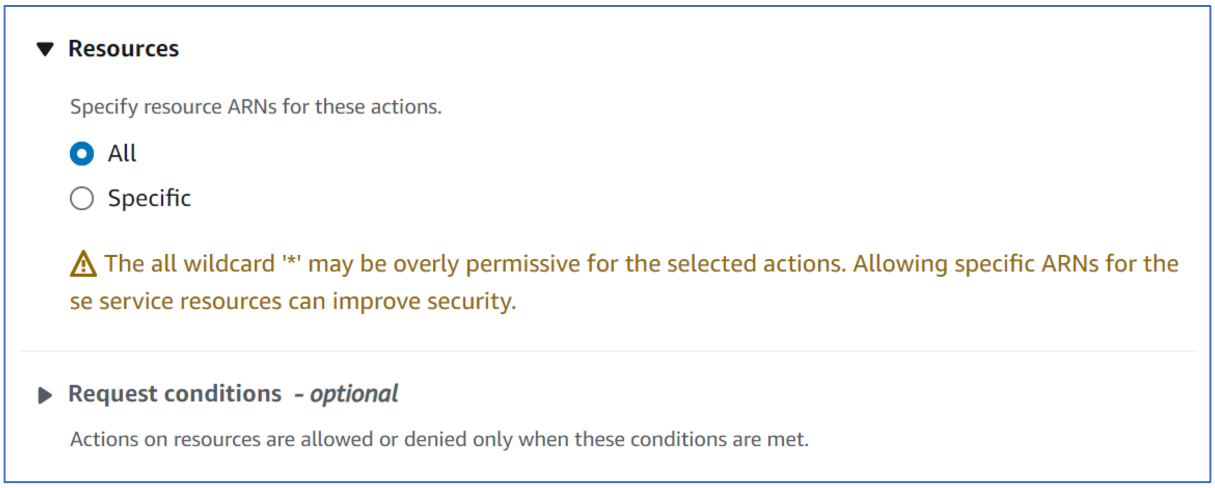
☑ AssumeRole Info

☐ AssumeRoleWithSAML Info

☐ AssumeRoleWithWebIdentity Info

☐ DecodeAuthorizationMessage Info

STEP 3: And then select all resources to create a inline policy for STS assume role to be attached to user:



Resources

Specify resource ARNs for these actions.

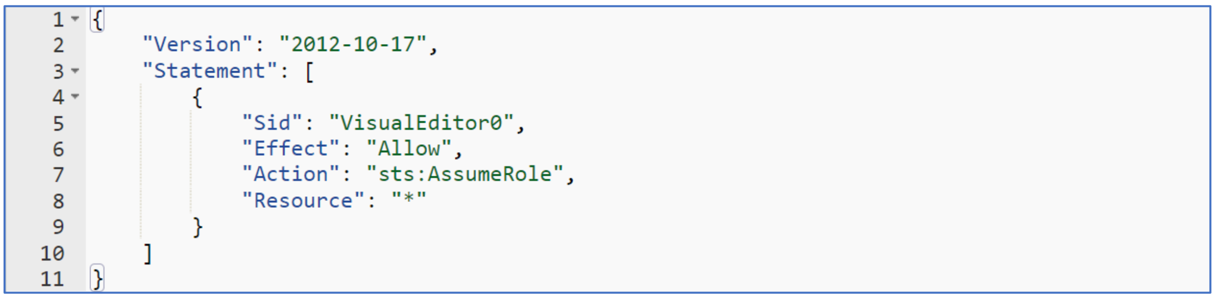
[✓ All] Specific

The all wildcard '*' may be overly permissive for the selected actions. Allowing specific ARNs for these service resources can improve security.

Request conditions - *optional*

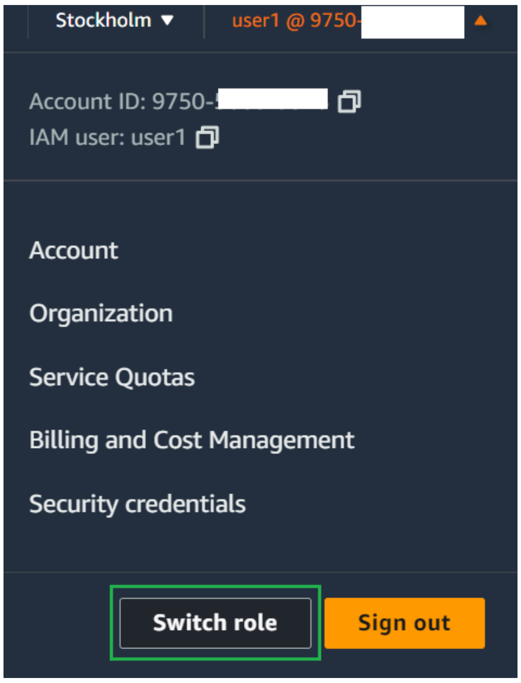
Actions on resources are allowed or denied only when these conditions are met.

Final JSON code will look like this: Now any user assigned with this policy can Assume Role.



```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "*"
        }
    ]
}
```

How to switch Role:



Stockholm ▼ | user1 @ 9750-[redacted] ▲

Account ID: 9750-[redacted]

IAM user: user1

Account

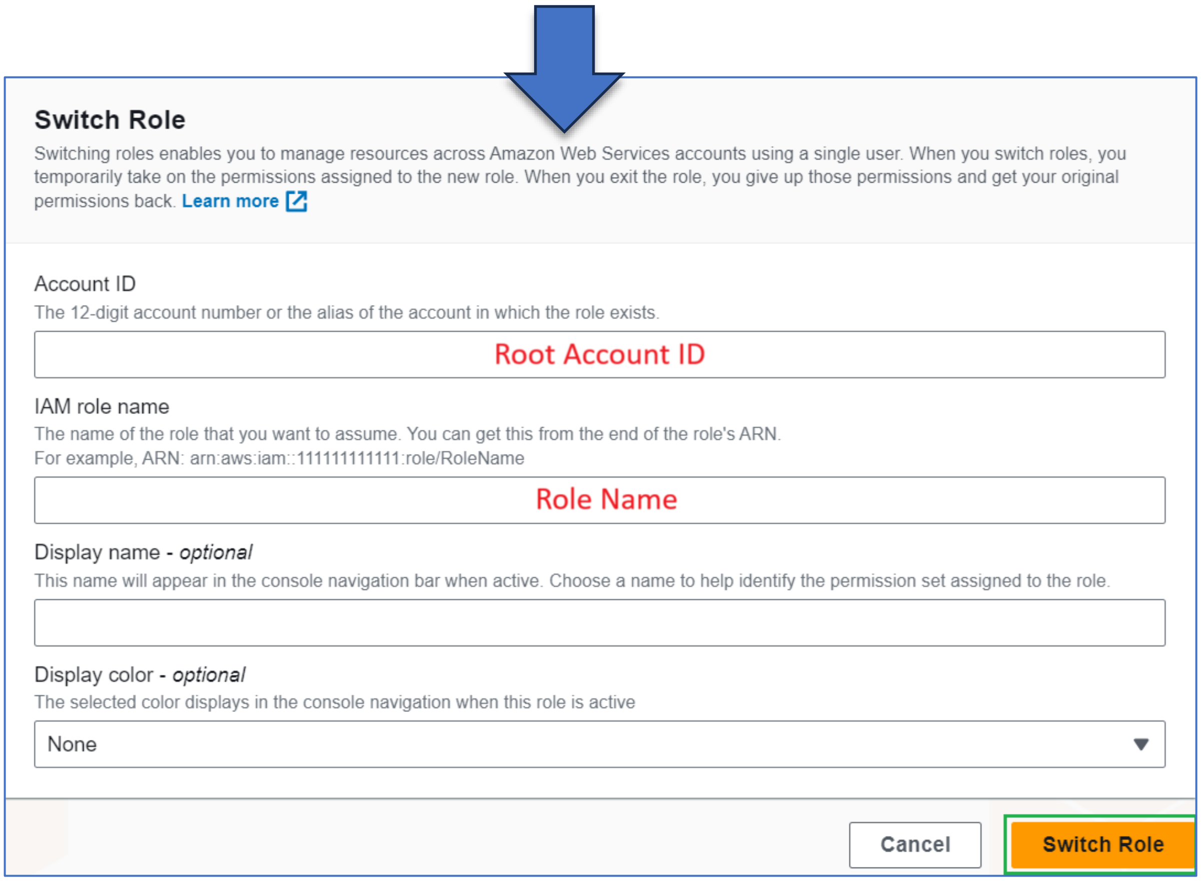
Organization

Service Quotas

Billing and Cost Management

Security credentials

Switch role | Sign out



Switch Role

Switching roles enables you to manage resources across Amazon Web Services accounts using a single user. When you switch roles, you temporarily take on the permissions assigned to the new role. When you exit the role, you give up those permissions and get your original permissions back. Learn more

Account ID

The 12-digit account number or the alias of the account in which the role exists.

Root Account ID

IAM role name

The name of the role that you want to assume. You can get this from the end of the role's ARN.
For example, ARN: arn:aws:iam::111111111111:role/RoleName

Role Name

Display name - *optional*

This name will appear in the console navigation bar when active. Choose a name to help identify the permission set assigned to the role.

Display color - *optional*

The selected color displays in the console navigation when this role is active

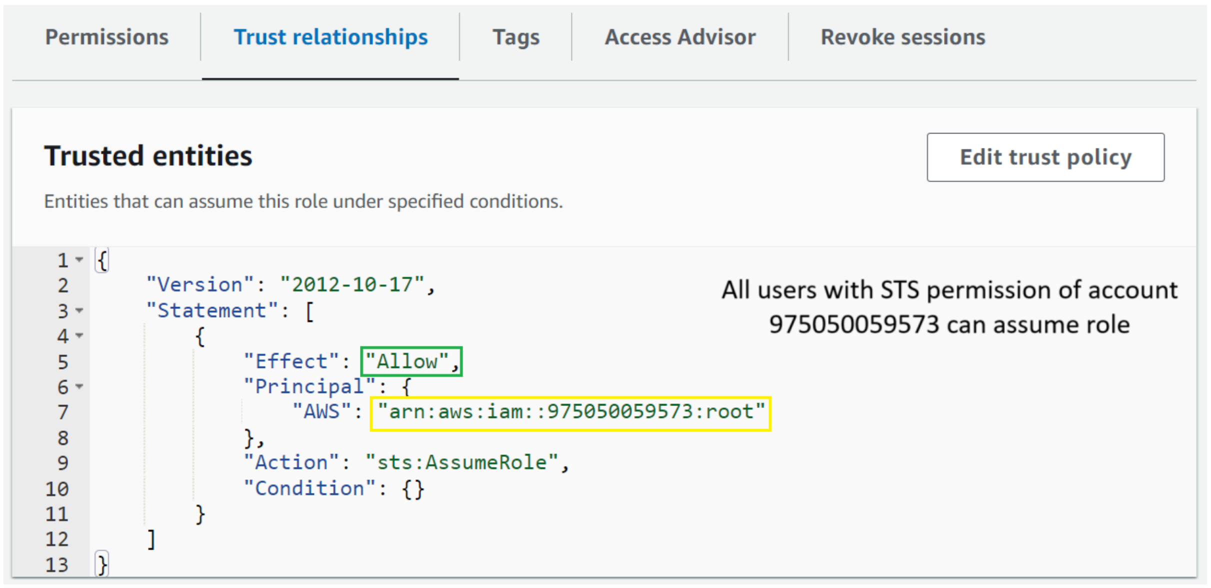
None ▼

Cancel | Switch Role

Use **Trust Relationship** to control access who can assume the role:

We can specify which entities can assume particular role under specified conditions.

For example below policy show all users in the account can assume role if STS permission is given:



Trusted entities

Entities that can assume this role under specified conditions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::975050059573:root"
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

All users with STS permission of account 975050059573 can assume role

To allow Role assume permission to specific users:

```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Sid": "one",
                        "Effect": "Allow",
                        "Principal": {
                                "AWS": [
                                  "arn:aws:iam::975050059573:user/user1",
                                  "arn:aws:iam::975050059573:user/user2"
                                  ]
                        },
                        "Action": "sts:AssumeRole",
                        "Condition": {}
                }
        ]
}
```

Here you can see that user1 and user2 can only assume the Role, rest all users are denied access to assume Role.

To deny the Role assume permission:

```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Sid": "one",
                        "Effect": "Allow",
                        "Principal": {
                                "AWS":
                                  "arn:aws:iam::975050059573:root"
                        },
                        "Action": "sts:AssumeRole",
                        "Condition": {}
                },
                {
                        "Sid": "two",
                        "Effect": "Deny",
                        "Principal": {
                                "AWS": "arn:aws:iam::975050059573:user/User3"
                        },
                        "Action": "sts:AssumeRole",
                        "Condition": {}
                }
        ]
}
```

In above trust relation policy you can see that all users in account 975050059573 are allowed to access the Role but User3 is denied so he user3 cannot assume role.

AWS Security Token Service (STS)

- The AWS Security Token Service (STS) is a web service that enables you to request temporary, limited-privilege credentials for IAM users or for users that you authenticate (federated users).
- By default, AWS STS is available as a global service, and all AWS STS requests go to a single endpoint at https://sts.amazonaws.com
- You can optionally send your AWS STS requests to endpoints in any region (can reduce latency).
- Credentials will always work globally.
- STS supports AWS CloudTrail, which records AWS calls for your AWS account and delivers log files to an S3 bucket.
- Temporary security credentials work almost identically to long-term access key credentials that IAM users can use, with the following differences:
- Temporary security credentials are short-term.
- They can be configured to last anywhere from a few minutes to several hours.
- After the credentials expire, AWS no longer recognizes them or allows any kind of access to API requests made with them.
- Temporary security credentials are not stored with the user but are generated dynamically and provided to the user when requested.
- When (or even before) the temporary security credentials expire, the user can request new credentials, if the user requesting them still has permission to do so.
- Advantages of STS are:
- You do not have to distribute or embed long-term AWS security credentials with an application.

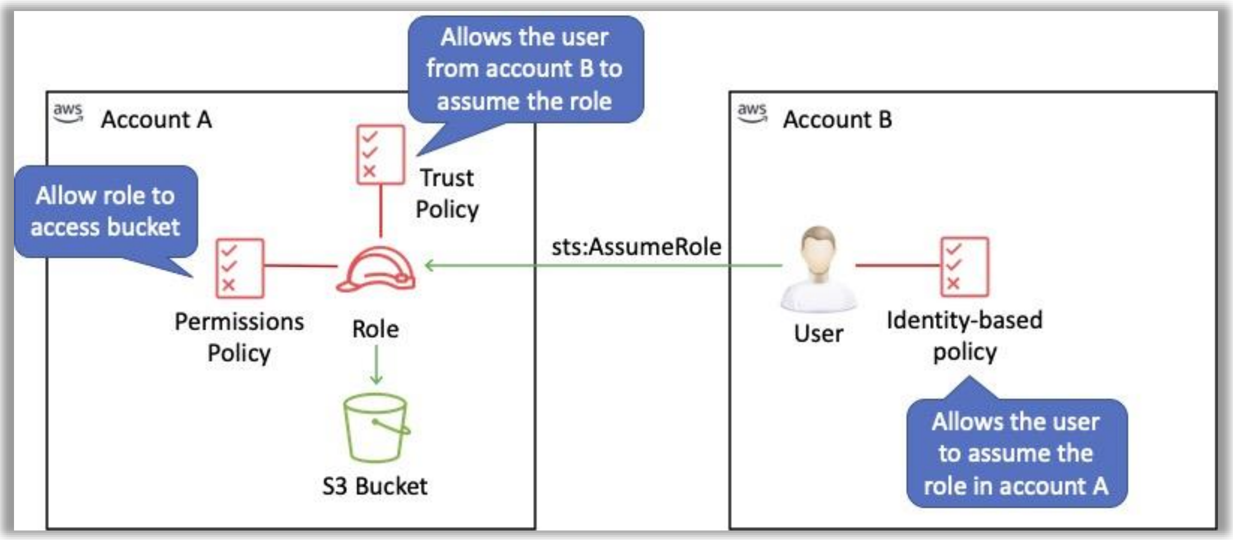
- You can provide access to your AWS resources to users without having to define an AWS identity for them (temporary security credentials are the basis for IAM Roles and ID Federation).
- The temporary security credentials have a limited lifetime, so you do not have to rotate them or explicitly revoke them when they're no longer needed.
- After temporary security credentials expire, they cannot be reused (you can specify how long the credentials are valid for, up to a maximum limit).
- The AWS STS API action returns temporary security credentials that consist of:
- An access key which consists of an access key ID and a secret ID.
- A session token.
- Expiration or duration of validity.
- Users (or an application that the user runs) can use these credentials to access your resources.

With STS you can request a session token using one of the following APIs:
AssumeRole – can only be used by IAM users (can be used for MFA).
AssumeRoleWithSAML – can be used by any user who passes a SAML authentication response that indicates authentication from a known (trusted) identity provider.
AssumeRoleWithWebIdentity – can be used by an user who passes a web identity token that indicates authentication from a known (trusted) identity provider.
GetSessionToken – can be used by an IAM user or AWS account root user (can be used for MFA).
GetFederationToken – can be used by an IAM user or AWS account root user.

Does not need to be a user in IAM.
Single sign-on allows users to login to the AWS console without assigning IAM credentials.

Cross Account Access:

- Lets users from one AWS account access resources in another.
- To make a request in a different account the resource in that account must have an attached resource-based policy with the permissions you need.
- Or you must assume a role (identity-based policy) within that account with the permissions you need.
- There are a couple of ways STS can be used.
- Useful for situations where an AWS customer has separate AWS account – for example for development and production resources.
- Cross Account Access makes is easier to work productively within a multi-account (or multi-role) AWS environment by making is easy to switch roles within the AWS Management Console.
- Can sign-in to the console using your IAM user name and then switch the console to manage another account without having to enter another user name and password.
- Let's users from one AWS account access resources in another.
- To make a request in a different account the resource in that account must have an attached resource-based policy with the permissions you need.
- Or you must assume a role (identity-based policy) within that account with the permissions you need.


Account A
Allows the user from account B to assume the role
Trust Policy
Allow role to access bucket
Permissions Policy
Role
S3 Bucket
sts:AssumeRole
Account B
User
Identity-based policy
Allows the user to assume the role in account A