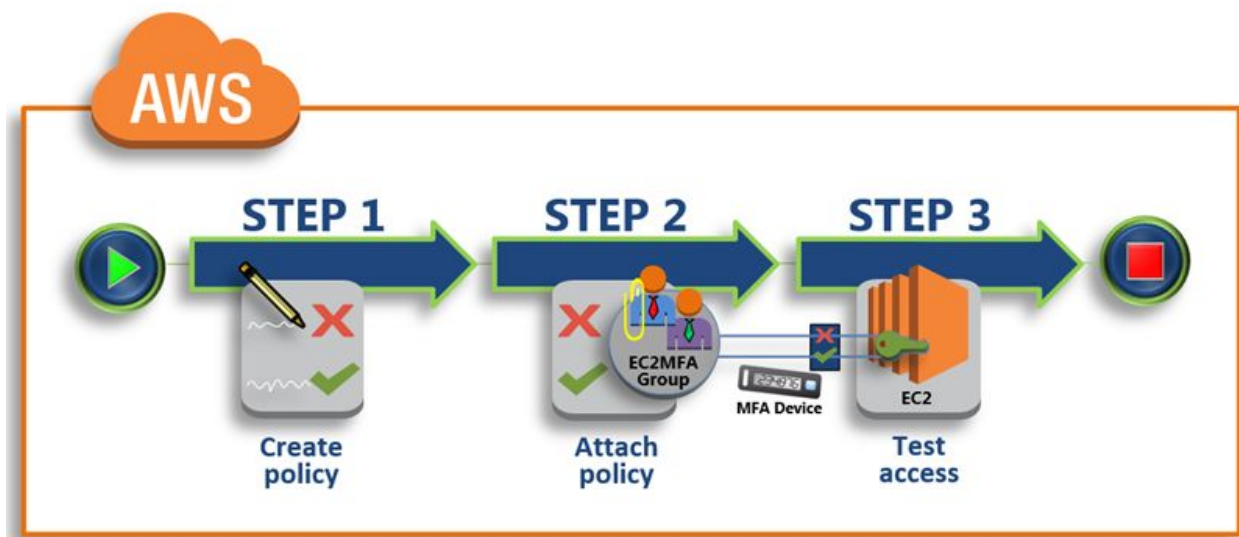


Enable Your Users to Configure Their Own Credentials and MFA Settings

You can enable your users to self-manage their own multi-factor authentication (MFA) devices and credentials. You can use the AWS Management Console to configure credentials (access keys, passwords, signing certificates, and SSH public keys) and MFA devices for your users in small numbers, but that is a task that could very quickly become time consuming as the number of users grows. Security best practice specifies that users should regularly change their passwords and rotate their access keys, delete or deactivate credentials that are not needed, and that they should use MFA, at the very least for sensitive operations. Showing you how to enable these best practices without burdening your administrators is the goal of this tutorial.

This tutorial shows how to grant users access to AWS services, but **only** when they sign in with MFA. If they are not signed in with MFA, then they are denied all permissions except those required to change their password or manage their MFA devices, including provisioning a new one.

This workflow has three basic steps.



Step 1: Create a Policy to Enforce MFA Sign-in

Create a customer managed policy that prohibits all actions **except** the few IAM APIs that enable changing credentials and managing MFA devices.

Step 2: Attach Policies to Your Test Group

Create a group whose members have full access to all Amazon EC2 actions if they sign-in with MFA by attaching both the AWS managed policy called `AmazonEC2FullAccess` and the customer managed policy you created in the first step.

Step 3: Test Your User's Access

Sign in as the test user to verify that access to Amazon EC2 is blocked *until* the user creates an MFA device and then signs in using that device.

Prerequisites

To perform the steps in this tutorial, you need to already have the following:

- An AWS account that you can sign in to as an IAM user with administrative permissions.
- Your account ID number which you'll type into the policy in Step 1.
- To find your account ID number, on the navigation bar at the top of the page, choose **Support** and then choose **Support Center**. You can find your account ID under this page's **Support** menu.
- A test IAM user with group membership as follows:

Create user account		Create and configure group account	
User Name	Other instructions	Group Name	Add user as a member
MFAUser	Clear the check box for Generate an access key for each user	EC2MFA	MFAUser

Step 1: Create a Policy to Enforce MFA Sign-in

You begin by creating an IAM customer managed policy that denies all permissions except those required for IAM users to manage their own credentials and MFA devices.

1. Sign in to the AWS Management Console as a user with administrator credentials. To adhere to IAM best practices, don't sign in with your AWS account root user credentials.
2. Open the IAM console at <https://console.aws.amazon.com/iam/>.
3. In the navigation pane, choose **Policies**, and then choose **Create Policy**.
4. On the **Create Policy** page, choose **Select** next to **Create Your Own Policy**.
5. On the **Review Policy** page, for **Policy Name** type **Force_MFA**, and for **Description** type something like **This policy allows users to manage their own passwords and MFA devices but nothing else unless they authenticate with MFA.**
6. In the **Policy Document** editor window, paste the following policy text, replacing all of the occurrences of *accountid* with your actual account ID number.

7. Copy

```
8. {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllUsersToListAccounts",
      "Effect": "Allow",
      "Action": [
        "iam:ListAccountAliases",
        "iam:ListUsers",
        "iam:GetAccountSummary"
      ],
      "Resource": "*"
    },
    {
      "Sid":
"AllowIndividualUserToSeeAndManageTheirOwnAccountInformation",
      "Effect": "Allow",
      "Action": [
```

```

        "iam:ChangePassword",
        "iam:CreateAccessKey",
        "iam:CreateLoginProfile",
        "iam:DeleteAccessKey",
        "iam:DeleteLoginProfile",
        "iam:GetAccountPasswordPolicy",
        "iam:GetLoginProfile",
        "iam:ListAccessKeys",
        "iam:UpdateAccessKey",
        "iam:UpdateLoginProfile",
        "iam:ListSigningCertificates",
        "iam:DeleteSigningCertificate",
        "iam:UpdateSigningCertificate",
        "iam:UploadSigningCertificate",
        "iam:ListSSHPublicKeys",
        "iam:GetSSHPublicKey",
        "iam:DeleteSSHPublicKey",
        "iam:UpdateSSHPublicKey",
        "iam:UploadSSHPublicKey"
    ],
    "Resource":
"arn:aws:iam::accountid:user/${aws:username}"
    },
    {
        "Sid": "AllowIndividualUserToListTheirOwnMFA",
        "Effect": "Allow",
        "Action": [
            "iam:ListVirtualMFADevices",
            "iam:ListMFADevices"
        ],
        "Resource": [
            "arn:aws:iam::accountid:mfa/*",
            "arn:aws:iam::accountid:user/${aws:username}"
        ]
    },
    {
        "Sid": "AllowIndividualUserToManageTheirOwnMFA",
        "Effect": "Allow",
        "Action": [
            "iam:CreateVirtualMFADevice",
            "iam:DeactivateMFADevice",
            "iam:DeleteVirtualMFADevice",
            "iam:RequestSmsMfaRegistration",
            "iam:FinalizeSmsMfaRegistration",
            "iam:EnableMFADevice",

```

```

        "iam:ResyncMFADevice"
    ],
    "Resource": [
        "arn:aws:iam::accountid:mfa/${aws:username}",
        "arn:aws:iam::accountid:user/${aws:username}"
    ]
},
{
    "Sid":
"BlockAnyAccessOtherThanAboveUnlessSignedInWithMFA",
    "Effect": "Deny",
    "NotAction": "iam:*",
    "Resource": "*",
    "Condition": { "BoolIfExists": {
"aws:MultiFactorAuthPresent": "false" } }
}
]
}

```

9. What does this policy do?

- The first statement enables the user to see basic information about the account and its users in the IAM console. These three permissions need to be in their own statement because they do not support or do not need to specify a specific resource ARN, and instead specify "Resource" : "*".
- The second statement enables the user to manage his or her own user, password, access keys, signing certificates, SSH public keys, and MFA information in the IAM console, including creating and changing them. The resource ARN limits the use of these permissions to only the user's own IAM user entity.
- The third statement enables the user to see information about MFA devices, and which are associated with his or her IAM user entity.
- The fourth statement allows the user to provision or manage his or her own MFA device. Notice that the resource ARNs in the fourth statement allow access to only an MFA device or user that has the exact same name as the currently signed-in user. Users can't create or alter any MFA device other than their own.

- The final statement uses a combination of "Deny" and "NotAction" to explicitly deny any non-IAM actions *if* the user is not signed-in with MFA. If the user is signed-in with MFA, then the "Condition" test fails and the final "deny" statement has no effect and other permissions granted to the user can take effect. This last statement ensures that when the user is not signed-in with MFA that they can only perform only the IAM actions allowed in the first four statements. The `...IfExists` version of the `Bool` operator ensures that if the user accesses an API with long-term credentials such as an access key, and the `aws:MultiFactorAuthPresent` key itself is therefore missing, the condition returns true and the statement still applies and the user is denied access to the non-IAM API.

10. Choose **Validate Policy** to ensure it is correct. Remember to replace your account ID for the placeholder above. After it passes validation, choose **Create Policy**.

Step 2: Attach Policies to Your Test Group


Next you attach two policies to the test IAM group which will be used to grant the MFA-protected permissions.

1. In the navigation pane, choose **Groups**.
2. In the search box, type **EC2MFA**, and then select the group in the list.
3. Choose the **Group Actions** button, and then chose **Attach**.
4. Choose the **Permissions** tab, and then click **Attach Policy**.
5. On the **Attach Policy** page, in the search box, type **EC2Full** and then select the check box next to **AmazonEC2FullAccess** in the list. Don't save your changes yet.
6. In the search box, type **Force**, and then select the check box next to **Force_MFA** in the list.

7. Choose **Attach Policy**.

Step 3: Test Your User's Access

In this part of the tutorial you sign in as the test user and verify that the policy works as intended.

1. Sign in to your AWS account as **MFAUser** with the password you assigned in the previous section. Use the URL: `https://<alias or account ID number>.signin.aws.amazon.com/console`
2. Choose **EC2** to open the Amazon EC2 console and verify that the user has no permissions to do anything.
3. On the navigation bar, choose **Services**, and then choose **IAM** to open the IAM console.
4. In the navigation pane, choose **Users**, and then choose the user (not the check box) **MFAUser**. If the **Groups** tab appears by default, note that it says you are not a member of any group. This is because you don't (in this scenario) have permissions to see your group memberships.
5. Now add an MFA device. Choose the **Security credentials** tab. Next to **Assigned MFA device**, choose the edit icon ().
6. For this tutorial, we'll use text messaging-based SMS MFA. Choose **An SMS MFA device**, and then click **Next Step**.
7. Enter your cell phone number and then choose **Next Step**. When you get the confirmation code as a text message on your phone, type the code into the box, choose **Activate SMS MFA**, and then choose **Finish**. You can now use MFA to sign in as this user.
8. Sign out of the console and then sign in as **MFAUser** again. This time AWS prompts you for an MFA code from your phone. When you get it, type the code in the box and then choose **Submit**.

9. Choose **EC2** to open the Amazon EC2 console again, and note that this time you can see all the information and perform any actions you wish. If you go to any other console as this user, you will see "access denied" messages because the policies in this tutorial grant access only to Amazon EC2.