**IAM**

**Introduction to AWS Identity and Access Management (IAM)**

Introduction

AWS Identity and Access Management (IAM) is a service that allows AWS customers to manage users' access and permissions to the AWS accounts and available APIs/services within AWS.

IAM can manage users, security credentials (such as API access keys), and allow users to access AWS resources.

In this lab, we'll walk through the foundations of IAM, focusing on user and group management, as well as assigning access to specific resources using IAM-managed policies.

We’ll log in as different AWS users and see how permissions that are assigned via our groups affect our users. And we’ll take a look at how managed access control list policies help us define permissions (e.g., access keys, passwords, etc.) at a granular level for our users that are members of individual groups. (This is particularly important nowadays when it comes to security and compliance issues.)

In AWS IAM, there's the root access account level, and, within that, we can create different groups to which we can add members. Groups provide different sets of permissions (also known as policies) for users, depending on what they require for their job function. Today, we'll get a firsthand look at how this all works.

Now, to get started, click the **Open Console** button that launched with the activity, and log in with the username cloud\_user and the password provided.

Environment Walkthrough

Under *AWS services* at the top, type "IAM." You can also find it by clicking **All services**and clicking **IAM** under the *Security, Identity, & Compliance* header.

It's important to note IAM is a global service on AWS (which you'll see noted in the top right corner of the dashboard here), which means these users and permissions will apply across all available AWS regions. As part of this activity, we’ll stop and start an EC2 instance that’s already been launched for us inside of the us-east-1 region. Instances are regional, but the permissions will apply regardless of where the instance is.

In the left sidebar, select the **Users** tab. There are four users already created, but we're going to focus on user-1, user-2, and user-3.

Click **user-1**. At the top, under *Summary*, we see the user’s ARN (Amazon Resource Name), path, and creation time. By selecting the **Permissions** and **Groups** tabs, we can see user-1 does not have any permissions assigned to it and does not belong to any groups. Select **Security credentials** to see its access keys, SSH public keys, and HTTPS Git credentials for AWS CodeCommit. Select the **Access Advisor** tab to see which services the user has accessed and when. We can see the other users' different permissions and groups, too, by clicking the **Users** tab in the left sidebar and checking out these same areas.

Now, let’s look at our groups. In the left sidebar, select **Groups**. There are three groups we're going to focus on:

* EC2-Admin: Provides permissions to view, start, and stop EC2 instances
* EC2-Support: Provides read-only access to EC2
* S3-Support: Provides read-only access to S3.

(We don’t need to worry about the cloud\_assessment\_labs group. If we opened this, we wouldn’t be able to view the permissions, since it's our lab environment limiting that access.)

We can click any of the groups here to see which policy is attached to it. There are two different kinds of policies for these groups:

* Managed policies: Policies shared among users and/or groups that are prebuilt either by AWS or an administrator within the AWS account. When it's updated, the changes to this policy are immediately applied for all users and groups to which it's attached.
* Inline policies: Policies assigned to just one user or group that are typically used in one-off situations.

Let's take a look at the EC2-Admin group, so go ahead and click that one. Once we select the **Permissions** tab, we can see it has a set of permissions associated with it: an inline policy. Now, click **Edit Policy** to see the actions the group is allowed to take (and which resources the action can be taken on) or if it has read-only access.

This policy displays JSON access control policy language and provides access on a granular level to AWS resources. For example, members of the EC2-Admin group have permissions to perform these actions, which map back to programmatic or API functions or calls. According to this policy, users belonging to this group can view all of the instances, as well as start and stop instances. Those are the actions they're allowed to take, and then below that, we can see the effects. If it said "Effect": "Deny", we wouldn’t be allowed to perform any of these actions.

(*Note:* It isn’t pertinent to this lab in particular, but something important to remember going forward — whether you continue with certifications or just get deeper into AWS — is that an explicit Deny will always overrule an Allow. So if there is a Deny of permissions on any policy that applies to a user, if a user belongs to multiple groups or has multiple policies applying to it, Deny will always overrule any Allow. And, by default, everything is a Deny, just not explicitly. We have to Allow permissions for the user to do anything.)

The "Resource" defines what resources we’re allowed to take these actions on. Since it has a \*, it means any EC2 resources. The EC2 service allows resource-level permissions, which means you could provide start and stop permissions to a single EC2 instance if you wanted to get that granular.

So, from this, we can see we have permission to view all of our elastic load balancers, list metrics, get metric statistics, and describe metrics (which our CloudWatch metrics automatically configured with our EC2 instance). The same permissions apply to our autoscaling service. All of these allow us to do it on any resource.

Now, click **Cancel**, select **Groups** from the left sidebar, and select EC2-Support. Under the **Permissions** tab, you'll notice it has a managed policy, which was already created by AWS. By clicking **Show Policy**, we'll see this group can describe EC2 instances, describe elastic load balancers, describe and list CloudWatch metrics, and describe our autoscaling configurations. What it doesn’t allow us to do is stop, start, create, or pretty much anything else. It’s essentially read-only, which means we can view what’s happening inside EC2, but we can’t do anything inside it.

Click **Cancel**, select **Groups** in the left sidebar one more time, and then select S3-Support. In this scenario, our S3-Support group is only allowed read-only access. Select **Show Policy**, where we'll see the Get and List actions, which allow us to view the objects in an S3 bucket as well as view the S3 bucket itself. Click **Cancel**, and let's move on to the next step.

Adding Users to Groups and Testing

Now, we want to assign users access to specific resources based on their job functions. You always want to follow the best practice of least privilege, meaning you should only provide the bare minimum permissions that are absolutely necessary for someone to perform their job. Here, we’re going to assign:

* user-1 to the S3-Support group
* user-2 to the EC2-Support group
* user-3 to the EC2-Admin group

We should still be on the S3-Support group page. Click the **Users** tab, and then hit the **Add Users to Group** button.

(*Note:* Once the users list appears, we may see what look like error messages. These aren’t error messages, though — they’re warnings that specific permissions aren’t allowed. As part of this lab, we use access control lists much like we’re learning here. This doesn’t impact our ability to move forward, so we don’t have to worry about it.)

Select user-1, and hit the **Add Users** button. user-1 is now part of the S3-Support group.

Now, we're going to repeat these steps for the remaining groups.

Click the **Groups** tab, and select EC2-Support. Click the **Users** tab, and then hit **Add Users to Group**. Select user-2, and hit **Add Users**, making user-2 a part of the EC2-Support group.

Finally, click the **Groups** tab, and select EC2-Admin. Click the **Users** tab, and then hit **Add Users to Group**. Select user-3, and hit **Add Users**. Now user-3 is part of the EC2-Admin group.

Now, let’s log in as these users to see their different permissions firsthand. We need to log in to the AWS console for this account as these users, so let’s find the login URL for them.

Click **Dashboard**, and we can see the *IAM users sign-in* link at the top. Click the double-papers icon to copy it, and then open an incognito-mode browser and paste the link into the browser. You could also copy and paste the link into your current browser, but this will log you out of the current cloud\_user account. (*Note:* Your *IAM users sign-in* link will be different than the one displayed in the lab video.)

Once you have the login page open in your chosen browser, enter the IAM username user-1 and the default password 123456, and then click **Sign In**. (*Note:* It goes without saying, but just in case: Do not use that password anywhere else, ever.) Under *AWS services* at the top, type "S3." (S3 can also be found by clicking **All services** and clicking **S3** under the *Security* header.)

user-1 has S3 read-only access, so we will not be able to create buckets or view EC2 resources when we’re logged in as user-1. We don’t have any buckets here, though, so we won't receive any error messages about being able to view it. However, we can test this permission (or lack of it) by clicking **+ Create bucket**, typing in a random bucket name (like, say, type-in-a-bucket-name), and clicking **Next** on each of the following screens up until we are finally able to click **Create bucket**. Then — sure enough — we get an "Error: Access denied" message. Click the **X** in the top right of the pop-up to close out of it.

Further, if we click **Services** in the top navigation bar, and then click **EC2** under the **Compute** header, we'll find we don’t have the authorized permissions to view any information here because this user (i.e., this particular team member inside our work environment) doesn’t need these permissions for their job role. Basically, we see a lot of "You are not authorized..." messages here.

Let's log out by clicking **user-1** in the top right corner, and then click **Sign Out**. This brings us to the main AWS page, where we can either click the **Sign In to Console**button in the top right corner, or paste our IAM users sign-in link back into the browser.

Our account ID should still be there, so all we need to do is enter user-2 as the IAM username and the password 123456, and then click **Sign In**. user-2 is part of the EC2-Support group, meaning it has read-only access to EC2. Navigate to EC2 by selecting **EC2** from the *Recently visited services* list, searching for it in the *Find Services* box like we did before, or clicking **All services** and clicking **EC2** under the *Compute* header.

*Important:* Make sure the region in the top right corner is *N. Virginia*, or else the lab will not work as intended.

We should immediately notice user-2 has more permissions than user-1 (i.e., no more "You are not authorized..." messages).

Under *Resources*, click **Running Instances**, and notice it has one instance running. This group does not have permission to stop instances, but let's see what it looks like if we try. Right-click the *running* status under the *Instance State* column for the instance listed. In the dropdown, hover over *Instance State* and click **Stop**. Upon clicking the **Yes, Stop** button in the *Stop Instances* pop-up, we'll receive a message that says "Error stopping instances." Click **Cancel** to close the pop-up.

Now, right-click the *running* status under the *Instance State* column for the instance listed, hover over *Instance State* in the dropdown, and click **Terminate**. Upon clicking the **Yes, Terminate** button in the *Terminate Instances* pop-up, we will receive a message that says "Error terminating instances." Click **Cancel** to close the pop-up.

Here's what we *can* do, though: With read-only permissions as user-2, we should be able to view the rest of EC2, including available load balancers and launch configurations, which we can access via the sidebar on the left.

Now, let's go back to Amazon S3, which worked for us a moment ago as user-1. It won't work for us now since user-2 has EC2-Support permissions and nothing else — unlike user-1, which had S3-Support permissions and nothing else. So now we see user-2 doesn't have any access here, as we receive an "Error: Access Denied" message.

Finally, let's log in as user-3. Click **user-2** in the top right corner, and then click **Sign Out**. Repeating the previous steps one last time, log in as user-3 with the password 123456, and click **Sign In**. Navigate to EC2, either under *Recently visited services*or by clicking **All services** and clicking **EC2** there.

As a reminder, as part of the EC2-Admin group, user-3 has permissions to view, start, and stop EC2 instances. Under *Resources*, click **Running Instances**, and right-click the *running* status under the *Instance State* column for the instance listed. In the dropdown, hover over *Instance State* and click **Stop**. This time, upon clicking the **Yes, Stop** button in the *Stop Instances* pop-up, the instance will begin moving into a stopping state. It may take a few minutes for it to finally stop. Once it’s stopped, we can then right-click the *stopped* status, hover over *Instance State* in the dropdown, click **Start**, and then click **Yes, Start** in the *Start Instances* pop-up. It will then show a *pending* status as it moves back into a *running* status.

Conclusion

Congratulations! You’ve completed this lab!

You now know how users can be assigned to groups, how permissions can be assigned to groups, and why it’s important to assign certain permissions only to specific users for their job functions. Continue your AWS journey by checking out some of our certification training, AWS Essentials, or AWS Concepts. Good luck out there!