Cloud Security with AWS IAM

Erik Gonzalez

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/Env": "development"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "ec2:DeleteTags",
        "ec2:CreateTags"
      ],
      "Resource": "*"
    }
  ]
}
```

Introducing Today's Project!

In this project, I'll show how to use AWS IAM to manage user access and permissions in an AWS account, building a foundation-level understanding of cloud security and how identities are securely controlled in the cloud.

Tools and concepts

Services I used were Amazon EC2 and AWS IAM. Key concepts I learnt include IAM users, policies, user groups, account aliases. Also learned how to use policy simulator and how JSON policies work. How to launch and tag an instance.

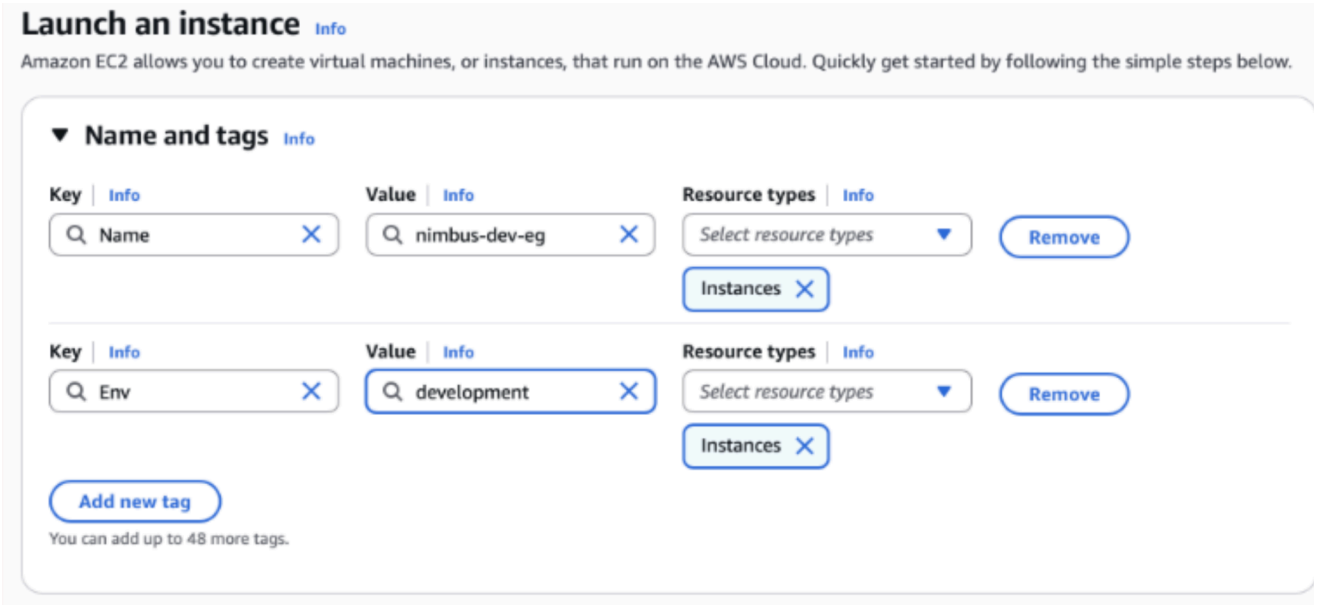
Project reflection

This project took me approximately 2 hours. The most challenging part was understanding the IAM policy since it was written in JSON and it contained multiple statements. It was most rewarding to see permission denied as the intern. Policy worked!

Tags

Tags are organizational tools that let us label our resources. They are helpful for grouping resources, cost allocations, and applying policies for all real resources with the same tag.

The tag I've used on my EC2 instances is called Env, which stands for environment. The values I've assigned for my instances are production and development.



IAM Policies

IAM policies are rules that can determine who can do what in a AWS account. To control who has access to my production/environment instance.

The policy I set up

For this project, I've set up a policy using JSON.

Made a policy that allows the policy holder (i.e. the intern) to have permission to do anything they desire to any instance tagged with "development." They can also see information for any instance, but they're denied access to deleting/creating tags

When creating a JSON policy, you have to define its Effect, Action and Resource.

The Effect, Action, and Resource attributes of a JSON policy means whether or not the policy is allowing or denying actions (i.e. effect); what the policy holder can or cannot do (i.e. the action) and specific AWS resources that the policy relates to

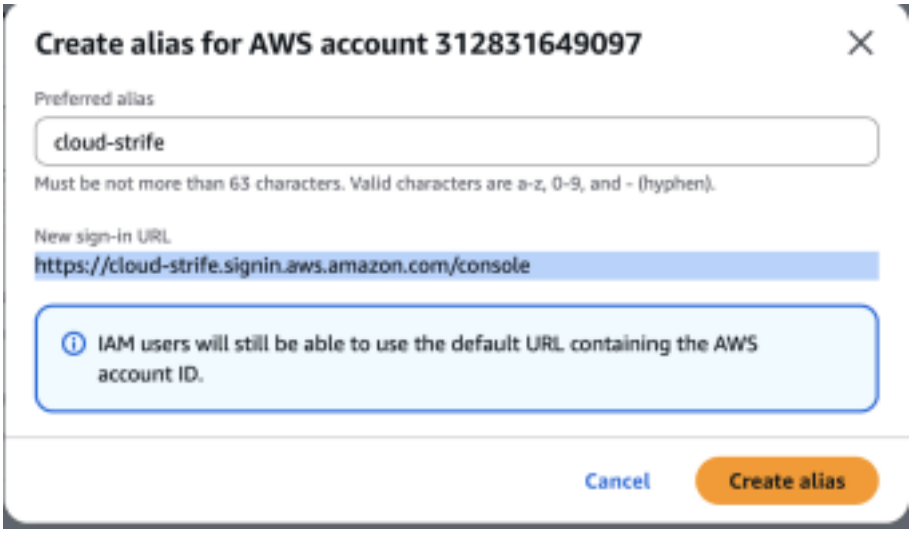
My JSON Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/Env": "development"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "ec2:DeleteTags",
        "ec2:CreateTags"
      ],
      "Resource": "*"
    }
  ]
}
```

Account Alias

An account alias is simply a nickname for the AWS account. Instead of a long account ID, we can now reference the account alias instead.

Creating an account alias took me less than a minute. The console login uses is https://cloud-strife.signin.aws.amazon.com/console



IAM Users and User Groups

Users

IAM users are people or entities that have access/can login to the AWS account.

User Groups

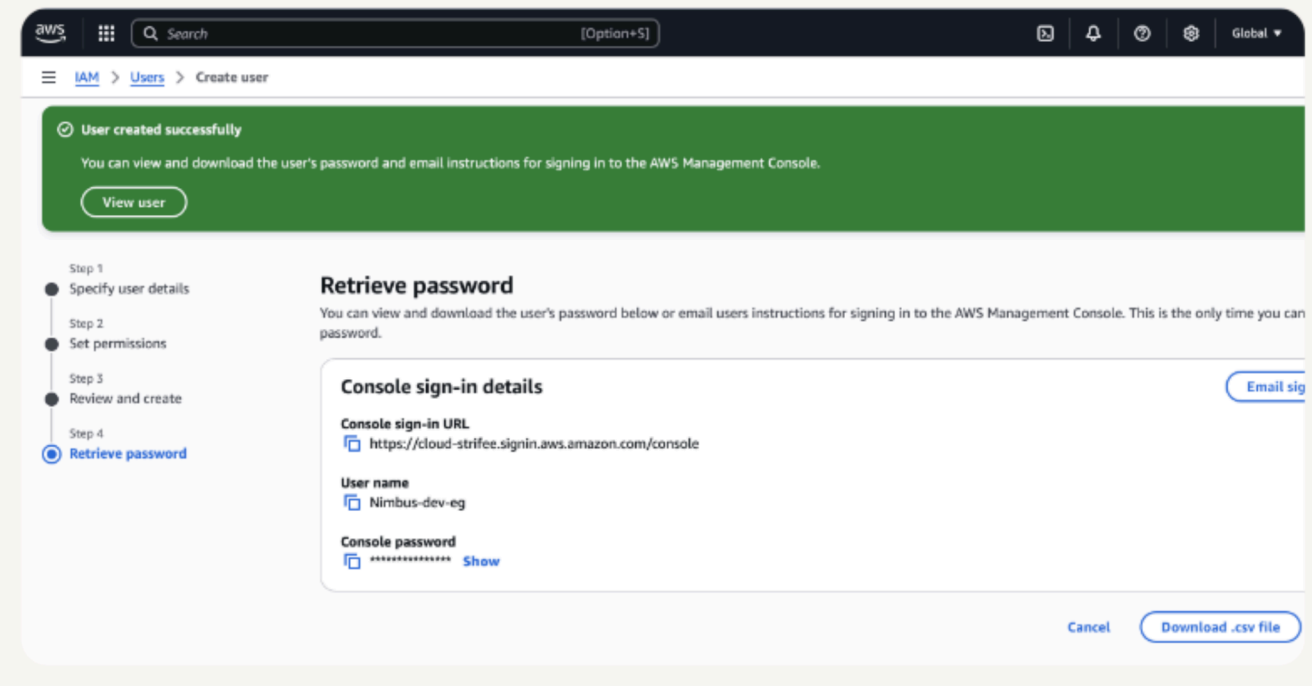
IAM user groups are like folders that collect IAM users so that you can apply permission settings at the group level.

I attached the policy I created to this user group, which means any user created inside this group will automatically get the permissions attached to our Nimbus Dev Environment IAM policy.

Logging in as an IAM User

The first way is to email the instructions, the second way is to download a .csv file with the sign-in details inside.

Once I logged in as my IAM user, I noticed that the intern user is already denied access to panels on the main AWS console dashboard. This was because we set up permissions in the EC2 instance to follow as such



Testing IAM Policies

I tested my JSON IAM policy by attempting to stop both the development and the production instances.

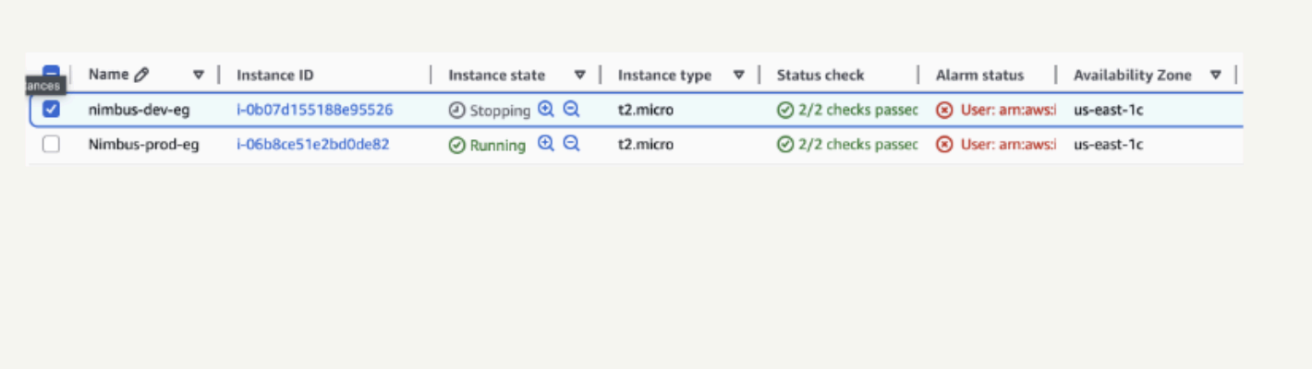
Stopping the production instance

When I tried to stop the production instance., I was met with an error. This was because the production instance is tagged with the 'production' label which is outside of the scope of the intern's permission policy. Intern is only granted development

Testing IAM Policies

Stopping the development instance

Next, when I tried to stop the development instance I successfully saw the instance state change to Stopping and then Stopped. This was because our permission policy allows the intern (users in dev group)) to stop instances.



IAM Policy Simulation

Why would you use the IAM Policy Simulator?

The IAM policy simulator is a tool that lets us simulate actions and test permission settings by defining a specific user/group/role and the action we want to test for. It's useful for saving time when testing permission settings. No more logging in into another user or actually stopping resources.

What were the simulation results for the development instance?

We set up a simulation for whether our dev group has permission to StopInstances or DeleteTags. The results were denied for both - we had to adjust the scope of the EC2 instances to ones that are tagged with “development”. Once we applied the tag, permission was allowed.