



Cloud Security with AWS IAM

N

Nigel Chitapi

The screenshot shows the AWS IAM 'Create policy' interface. On the left, a sidebar indicates 'Step 1: Specify permissions' is selected, with 'Step 2: Review and create' listed below it. The main area is titled 'Specify permissions' and contains a JSON editor. The JSON code is as follows:

```
1 v {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "ec2:Describe*",
7       "Resource": "*",
8       "Condition": {
9         "StringEquals": {
10           "ec2:ResourceTag/Env": "development"
11         }
12       }
13     },
14     {
15       "Effect": "Allow",
16       "Action": "ec2:Describe*",
17       "Resource": "*"
18     }
19 }
```

To the right of the JSON editor is a panel titled 'Edit statement' which says 'Select a statement'. It also includes a note: 'Select an existing statement in the policy or add a new statement.' A blue button labeled '+ Add new statement' is visible.

Introducing Today's Project!

Project overview

In this project, I will demonstrate how to use AWS IAM to control access and permission settings in my AWS account. I'm doing this project to learn cloud security from the basics and foundationtions. Every company thinks about access permissions, and there are even jobs called 'IAM Engineers" focused on these specifics security skills.

Tools and concepts

Services I used were EC2, IAM. Key concepts I learnt include IAM Users and Groups, IAM Policies, Instance creation, IAM Policy Simulator, Logging in as another user.

Project reflection

This project took me approximately 1.5hrs including demo time. The most challenging part was understand IAM JSON Policy structures and statement. It was most rewarding to see the IAM Policies work efficiently.

Tags

What I did in this step

In this step, I will launch 2 EC2 instances because i need to boost NextWork's computing power. The company is expecting more users and traffic into their website for a period of time over the summer break.

Understanding tags

Tags are organizational tools that enable us to label our resources.

My tag configuration

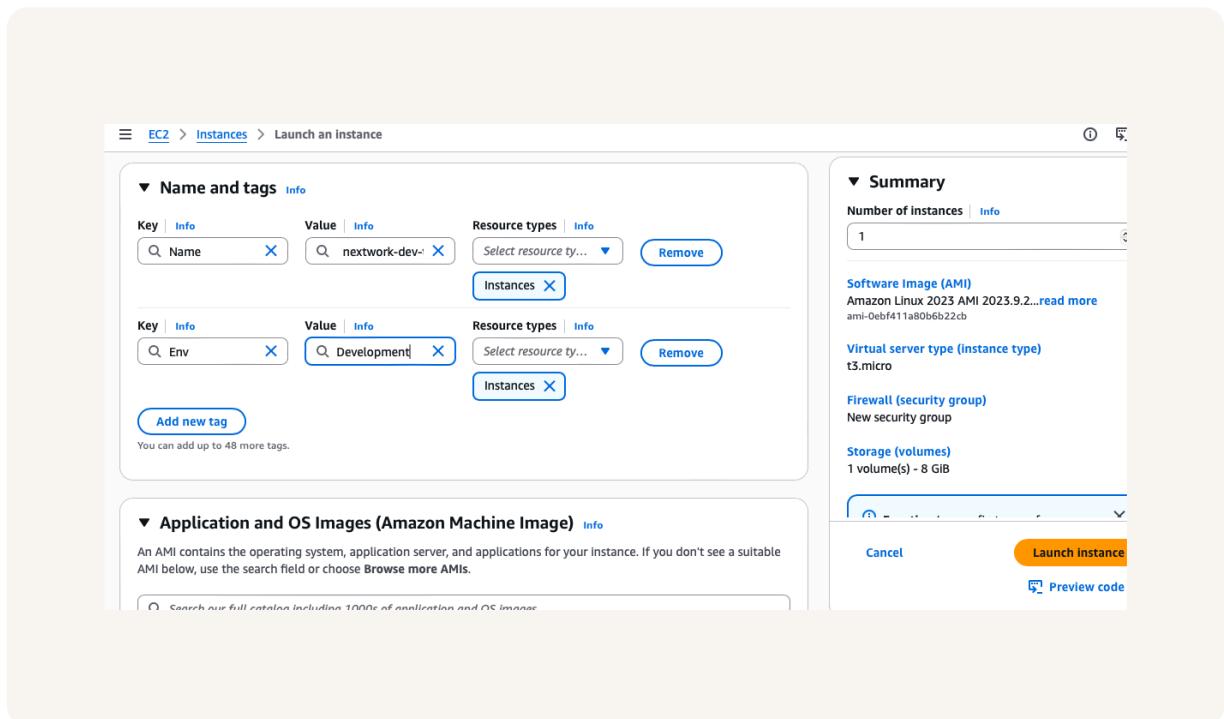
The tag I've used on my EC2 instances is called Environment. The values I've assigned for my instances are Production and Development.



Nigel Chitapi

NextWork Student

nextwork.org



IAM Policies

What I did in this step

In this step, I will use IAM Policies to control the access level of the newly hired intern. This is because they should only have access to the Development Environment and not the Production Environment.

Understanding IAM policies

IAM Policies are like rules that determine who can do what on AWS resources.

The policy I set up

For this project, I've set up a policy using JSON. I am using policies to control who has access to the production instance.

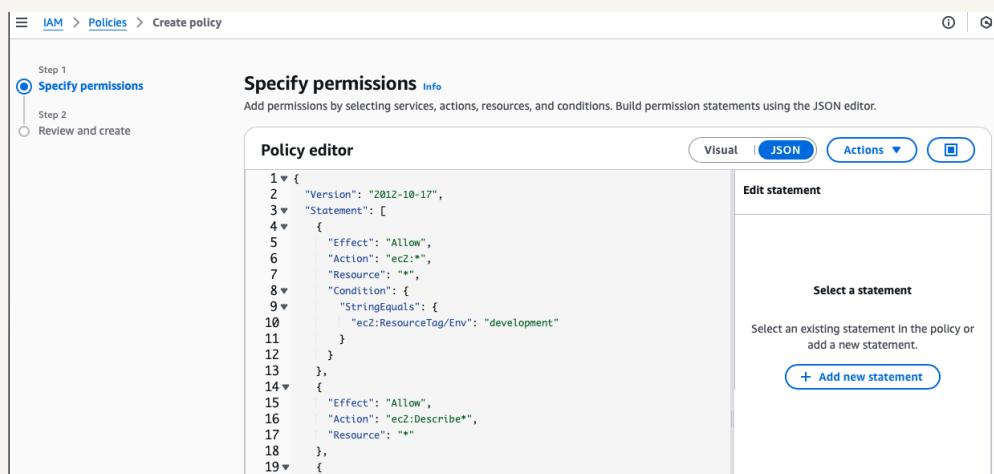
Policy effect

I've created a policy that allow the policy holder to have permission to do anything they want to any instance with the tag named "development". They can also see any information related to any instance however they can not delete or create tags for any instance

Understanding Effect, Action, and Resource

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

My JSON Policy



The screenshot shows the AWS IAM 'Create policy' wizard at Step 1: Specify permissions. The policy editor displays the following JSON code:

```
1 « {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": "ec2:*",
7             "Resource": "*",
8             "Condition": {
9                 "StringEquals": {
10                     "ec2:ResourceTag/Env": "development"
11                 }
12             }
13         },
14         {
15             "Effect": "Allow",
16             "Action": "ec2:Describe*",
17             "Resource": "*"
18         },
19     ]
}
```

The right panel of the editor is titled 'Edit statement' and contains a section for adding new statements with a button labeled '+ Add new statement'.

Account Alias

What I did in this step

In this step, I will setup an account alias which is like a nickname for an AWS account console log in. This is because an account alias makes it easy for our users to log-in

Understanding account aliases

An account alias is nickname for my AWS account instead of a long account ID we can now reference our account alias instead.

Setting up my account alias

Creating an account alias took me 15 seconds. Now, my new AWS console sign-in URL uses my alias and not my account ID.



The screenshot shows the AWS Identity and Access Management (IAM) service. A modal window titled "Create alias for AWS account 609604398881" is open. In the "Preferred alias" field, the value "nextwork-alias-tawanda" is entered. Below the field, a note states: "Must be not more than 63 characters. Valid characters are a-z, 0-9, and - (hyphen)." The "New sign-in URL" field contains the value "https://nextwork-alias-tawanda.siginin.aws.amazon.com/console". A tooltip in this field explains: "IAM users will still be able to use the default URL containing the AWS account ID." At the bottom right of the modal is a yellow "Create alias" button. The background shows the IAM dashboard with various statistics: 0 users, 0 groups, 13 policies, 7 roles, and 0 identities.

IAM Users and User Groups

What I did in this step

In this step, I will setup 2 resources IAM Users and IAM groups. This is because IAM Users are like log-ins for people who want access to our AWS account and the IAM User Groups are like folders to manage users with the same level of access.

Understanding user groups

IAM user groups are like folders that contain IAM Users so that we can apply permission policies at the User Group

Attaching policies to user groups

I attached the policy I created to this user group, which means every user assigned to this group will be bound by the policies attached to the group

Understanding IAM users

IAM users are people or entities that can access or log into the AWS account.

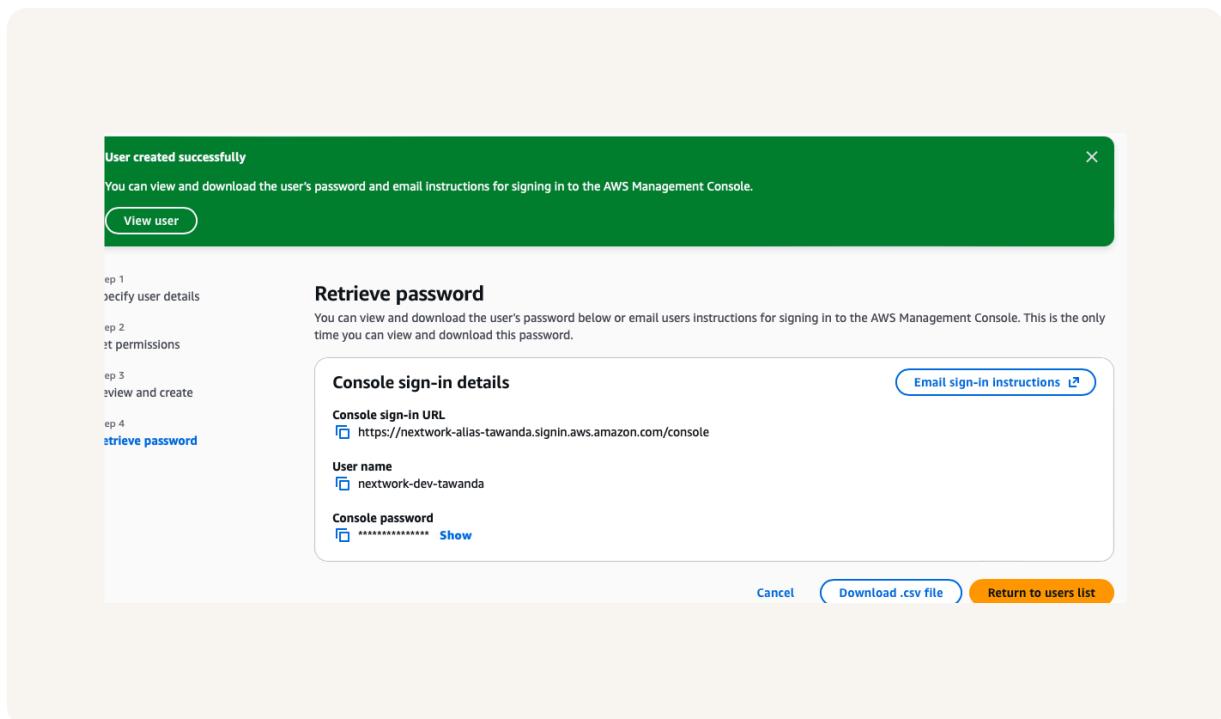
Logging in as an IAM User

Sharing sign-in details

The first way is email direct to the user or download the csv file

Observations from the IAM user dashboard

Once I logged in as my IAM user, I noticed that the intern user is denied access to panels on the main AWS Console dashboard. This was because we only setup permissions to the Development EC2 instance, so the intern would not have access to anything else or see anything else.



Testing IAM Policies

What I did in this step

In this step, I will log into the AWS account as the intern and test access to the production and development instances because I need to ensure that the intern does not have the ability to do anything that might affect the production environment

Testing policy actions

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



The screenshot shows the AWS CloudWatch Instances console. On the left, a sidebar lists navigation items: Instances, View (with a dropdown menu), iES, plates, its, is, instances, osts, eservations, nager, and New. The main area displays an error message for instance i-090bbfa8f2028347:

Failed to stop the instance i-090bbfa8f2028347
You are not authorized to perform this operation. User: arn:aws:iam::509604398881:user/network-dev-tawanda is not authorized to perform: ec2:StopInstances on resource: arn:aws:ec2:us-east-1:2609604398881:instance/i-090bbfa8f2028347 because no identity-based policy allows the ec2:StopInstances action.

Below the error message is a "Diagnose with Amazon Q" button. The main content area shows the instance details for i-090bbfa8f2028347:

Instance (1/1) [View details](#)

Last updated: [2024-01-16T10:45:22Z](#) | Connect | Instance state: [Running](#) | Actions | Launch instances

i-090bbfa8f2028347 (network-prod-tawanda)

[Details](#) | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary [Info](#)

Instance ID	Public IPv4 address	Private IPv4 addresses
i-090bbfa8f2028347	44.248.20.211 open address	172.31.14.155
IPv6 address	Instance state	Public DNS

Stopping the development instance

Next, when I tried to stop the development instance I successfully saw the instance state stopping. This was because the permission policy allows users in the Dev Users to perform any action on the Development EC2 instance

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with links for EC2, Dashboard, AWS Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, and Volumes.

The main content area displays the following information:

- Instances (1/2) Info**: Shows the last updated time as "less than a minute ago".
- Actions**: Includes "Launch instances" and other options.
- Table Headers**: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS.
- Table Data**:
 - nextwork-dev-... (Running, t3.micro, Initializing, User: armawski, us-west-2c, ec2-18-237-149-218.l)
 - nextwork-pro-... (Stopped, t3.micro, -, User: armawski, us-west-2c, -)
- Selected Instance Details**:
 - i-090bbfa8f20283437 (nextwork-pro-tawanda)**
 - Details** tab is selected, showing Status alarms, Monitoring, Security, Networking, Storage, and Tags.
 - Instance summary**:
 - Instance ID: i-090bbfa8f20283437
 - IPv6 address: -
 - Hostname type: IP name: ip-172-31-14-135.us-west-2.compute.internal
 - Public IPv4 address: 172.31.14.135
 - Private IP DNS name (IPv4 only): ip-172-31-14-135.us-west-2.compute.internal
 - Instance state: Stopped
 - Public DNS: -

IAM Policy Simulator

To extend my project, I'm going to test the permission policies in a safer and more controlled way using a tool named IAM Policy Simulator. I'm doing this because having to stop instances and log into AWS accounts as other users is quite inefficient and can be disruptive.

Understanding the IAM Policy Simulator

The IAM Policy Simulator is a tool that allows me to test permission settings by defining a specific user, group, role that I want to test for. It's useful for saving time when testing permissions.

How I used the simulator

I set up a simulation for whether the Development user group had permission to stop instances or delete tags. The results were Denied! (which was great!) I had to adjust the scope of the EC2 instances to ones that are tagged development and this time around the actions on the EC2 instances tagged Development were allowed.

The screenshot shows the IAM Policy Simulator interface. On the left, the policy document is displayed:

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

The right side shows the "Policy Simulator" results table:

Service	Action	Resource Type	Simulation Resource	Permission
Amazon EC2	StopInstances	instance	*	allowed 1 matching statements.
Amazon EC2	DeleteTags	not required	*	denied 1 matching statements.



nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects

