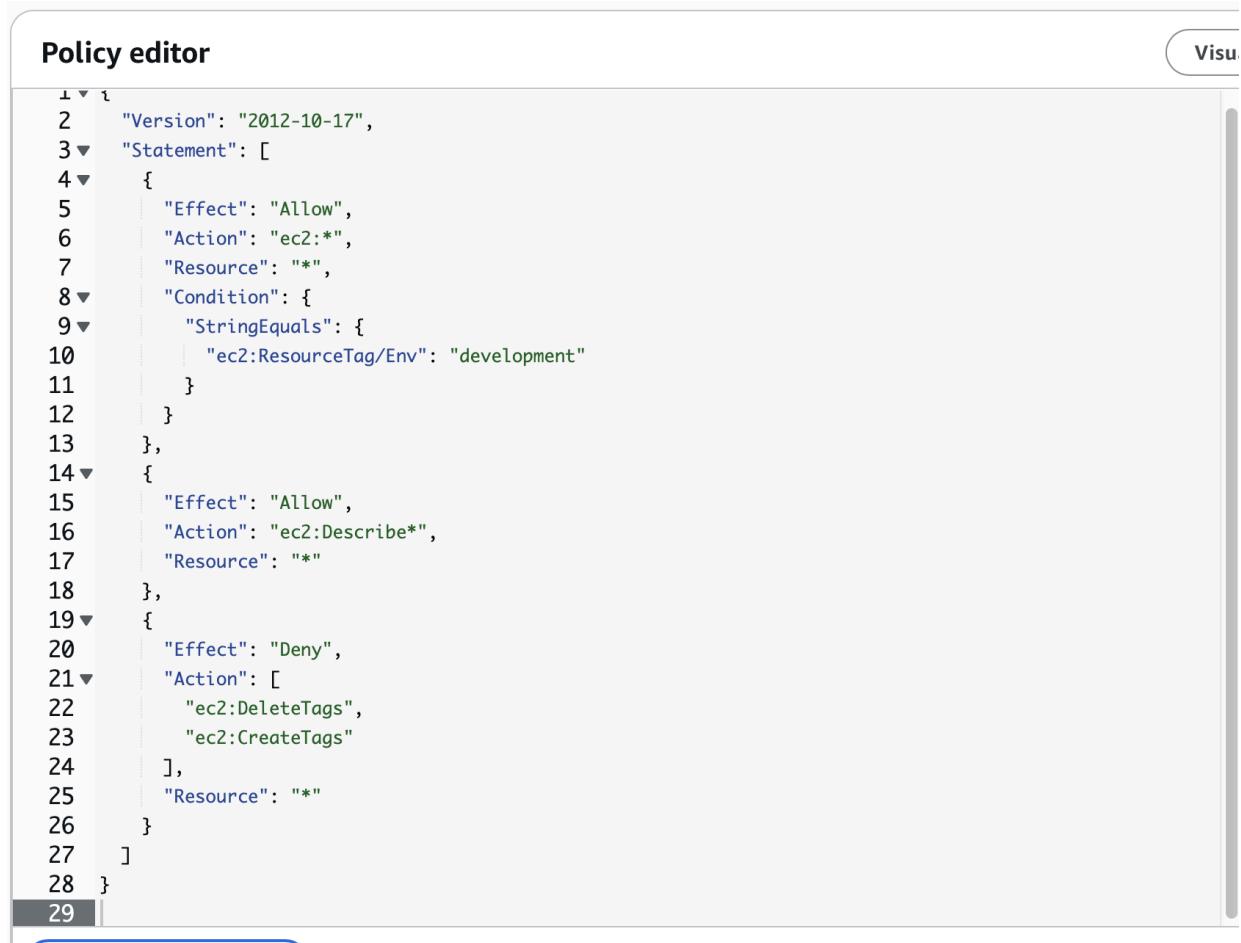


# Cloud Security with AWS IAM

Erik Gonzalez



The screenshot shows the AWS IAM Policy editor interface. The title bar says "Policy editor". On the right, there's a "Visual" button. The main area displays a JSON policy document with line numbers from 1 to 29 on the left. The policy grants various permissions for EC2 resources based on tags.

```
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        {
20            "Effect": "Deny",
21            "Action": [
22                "ec2:DeleteTags",
23                "ec2:CreateTags"
24            ],
25            "Resource": "*"
26        }
27    ]
28}
29
```

# 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. I also learned how to use a 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.

**Launch an instance** Info

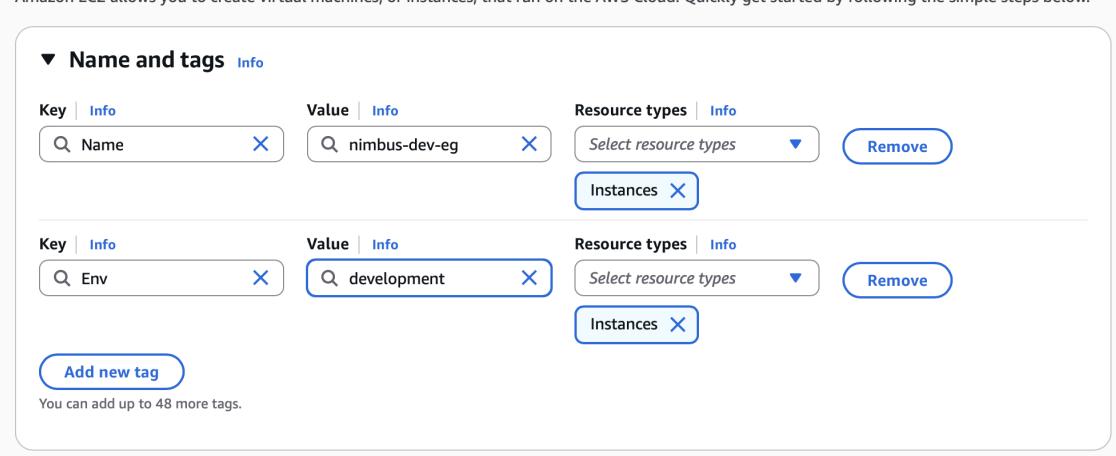
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

**▼ Name and tags** Info

<b>Key</b> <small>Info</small> <input type="text" value="Name"/> <span>X</span>	<b>Value</b> <small>Info</small> <input type="text" value="nimbus-dev-eg"/> <span>X</span>	<b>Resource types</b> <small>Info</small> <input type="text" value="Select resource types"/> <span>▼</span> <span>Remove</span>
<input type="text" value="Instances"/> <span>X</span>		
<b>Key</b> <small>Info</small> <input type="text" value="Env"/> <span>X</span>	<b>Value</b> <small>Info</small> <input type="text" value="development"/> <span>X</span>	<b>Resource types</b> <small>Info</small> <input type="text" value="Select resource types"/> <span>▼</span> <span>Remove</span>
<input type="text" value="Instances"/> <span>X</span>		

**Add new tag**

You can add up to 48 more tags.



# IAM Policies

IAM policies are rules that can determine who can do what in an 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

Policy editor

Visu

```
1 "Version": "2012-10-17",
2 "Statement": [
3     {
4         "Effect": "Allow",
5         "Action": "ec2:*",
6         "Resource": "*",
7         "Condition": {
8             "StringEquals": {
9                 "ec2:ResourceTag/Env": "development"
10            }
11        }
12    },
13    {
14        "Effect": "Allow",
15        "Action": "ec2:Describe*",
16        "Resource": "*"
17    },
18    {
19        "Effect": "Deny",
20        "Action": [
21            "ec2:DeleteTags",
22            "ec2:CreateTags"
23        ],
24        "Resource": "*"
25    }
26]
27]
28}
29
```

# 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>

### Create alias for AWS account 312831649097 X

Preferred alias

Must be not more than 63 characters. Valid characters are a-z, 0-9, and - (hyphen).

New sign-in URL

*ⓘ* IAM users will still be able to use the default URL containing the AWS account ID.

---

Cancel Create alias

# 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

The screenshot shows the AWS IAM 'Create user' success page. At the top, there's a navigation bar with the AWS logo, search bar, and global settings. Below it, the breadcrumb trail shows 'IAM > Users > Create user'. A green success message box contains the text 'User created successfully' and a note: 'You can view and download the user's password and email instructions for signing in to the AWS Management Console.' A 'View user' button is present. To the left, a vertical step-by-step guide shows 'Step 1 Specify user details', 'Step 2 Set permissions', 'Step 3 Review and create', and 'Step 4 Retrieve password', with 'Retrieve password' being the current step. On the right, under 'Console sign-in details', the 'Console sign-in URL' is listed as <https://cloud-strifee.signin.aws.amazon.com/console>. The 'User name' is listed as 'Nimbus-dev-eg'. The 'Console password' is shown as a masked string '\*\*\*\*\*' with a 'Show' link. There are 'Email sig' and 'Download .csv file' buttons at the bottom right.

# 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.

 Failed to stop the instance i-06b8ce51e2bd0de82

You are not authorized to perform this operation. User: arn:aws:iam::312831649097:user/Nimbus-dev-eg is not authorized to perform: ec2:StopInstances on resource: arn:aws:ec2:us-east-1:312831649097:instance/i-06b8ce51e2bd0de82 because no identity-based policy allows the ec2:StopInstances action. Encoded authorization failure message: DplPQrmA72z\_XvsjbAma9hldl2w9QMFt\_GtQvlrHP5ZKGiT\_Dd6CX5VnR9\_azpzULCaCNvbUXW965PyPuXXMMPqQ0Et8Ga3-1xQtpa1ZH1CJ7VAO7PsLpmgct29cZIAZgzfMtE9WDTw\_1Vjp2yS-DQjqUXCKR7OBdaeOHZdNxvn6bA7xeH2lNoV7Jgf1055Wuv7Eeg1k5tASJf0OkwrbI6s7UH2vbJaYBkQ4TTZqkTaqMMTqaYJji-dRZGUgwZyF09aPlu0x4w/41GBK8Tjncoix-w8AU-C8ofdhsO7HGB2zOgl\_2CR92KK\_lZC4kKM72qaBniNRdqLUiAX57NgjfZSiVELwjHHqzWI\_78m8rYSghEL8-DOoMMiV557t\_H9ZcfEwCEXhNNhP\_UrWt-Q6xTFb154XQVPJ8j2QPQBjP\_ifnw-OvzSGZWeoEvqj9vLb57ali21UP11uZXgFiozzHm-1z7H3cOkDx5J6VKG\_CR9pVdkvv3ps87nYTHwg7h12Ud4Fh81jniKEQsnYh6s9GbT6mGXJfx6EuxFI\_R95fsWMIM-cbURGG0JznixN79nVs4k5ctvMTdewEjyulKwMJs131Hizpi0Skvrg\_GkaaspL9R40X5gv8kqgzKZNajxAl4cor6JxA08Zbp7hgTqJWLcpnQ615P0\_6\_pT06am-TkpksamovS2U74x3xavsF6a1UN4tFm6EYZr-yen3FQ56cYg3HcRIC70CPW6Td7vJMDf5QHW7bf6A0zPs0XFdi\_RKT\_Zl75E24C4zUBfdJP5FcbsG0yW3iDYcNPvtIJ-hdKmgK7rbC7aDk2xtQHpg1q5xsTnat4OGGv0v7nYR0WNA0AO7E1wssg75tvPcY5-OFCwsTfhg

 Diagnose with Amazon Q X

# 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.

Instances	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input checked="" type="checkbox"/>	nimbus-dev-eg	i-0b07d155188e95526	Stopping	t2.micro	2/2 checks passed	User: arn:aws:iam::123456789012:root	us-east-1c
<input type="checkbox"/>	Nimbus-prod-eg	i-06b8ce51e2bd0de82	Running	t2.micro	2/2 checks passed	User: arn:aws:iam::123456789012:root	us-east-1c

# 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.

The screenshot shows the AWS IAM Policy Simulator interface. At the top, there is a navigation bar with tabs for 'Policy Simulator' (selected), 'AWS Lambda', and 'AWS Lambda Functions'. Below the navigation bar, there are dropdown menus for 'Service' (set to 'Amazon EC2') and 'Action(s) selected' (showing '2 Action(s) sele...'), along with buttons for 'Select All', 'Deselect All', 'Reset Contexts', 'Clear Results', and a prominent blue 'Run Simulation' button. A section titled 'Global Settings' is expanded, showing a note about matching statements. Below this is a table titled 'Action Settings and Results' with the following data:

Service	Action	Resource Type	Simulation Resource	Permission
Amazon EC2	StopInstances	Instance	*	<b>allowed</b> 1 matching statements.
Amazon EC2	DeleteTags	not required	*	<b>denied</b> 1 matching statements.