



Cloud Security with AWS IAM

D

Duc Thai

The screenshot shows the AWS IAM 'Create policy' interface. The left sidebar indicates 'Step 1: Specify permissions' is selected. The main area is titled 'Specify permissions' with a note: 'Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.' A large JSON code block is displayed, representing the policy definition. On the right, there's a 'Policy editor' window with tabs for 'Visual' (disabled), 'JSON' (selected), and 'Actions'. It includes a 'Select a statement' dropdown and a '+ Add new statement' button. The bottom of the screen shows the JSON code again with a character count of '5851 of 6144 characters remaining'.

```
1 * {
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 {
20     "Effect": "Deny",
21     "Action": [
22         "ec2:DeleteTags",
23         "ec2:CreateTags"
24     ],
25     "Resource": "*"
26 }
27
28 }
```

Introducing Today's Project!

Today I will be using the AWS Identity and Access Management (IAM) service to control who is authenticated (signed in) and authorized (has permissions) in my AWS account.

Tools and concepts

Services I used were IAM policy: JSON rules that allow or deny actions on resources. Account alias: a friendly account name used in the sign-in URL. IAM users: individual identities with credentials.

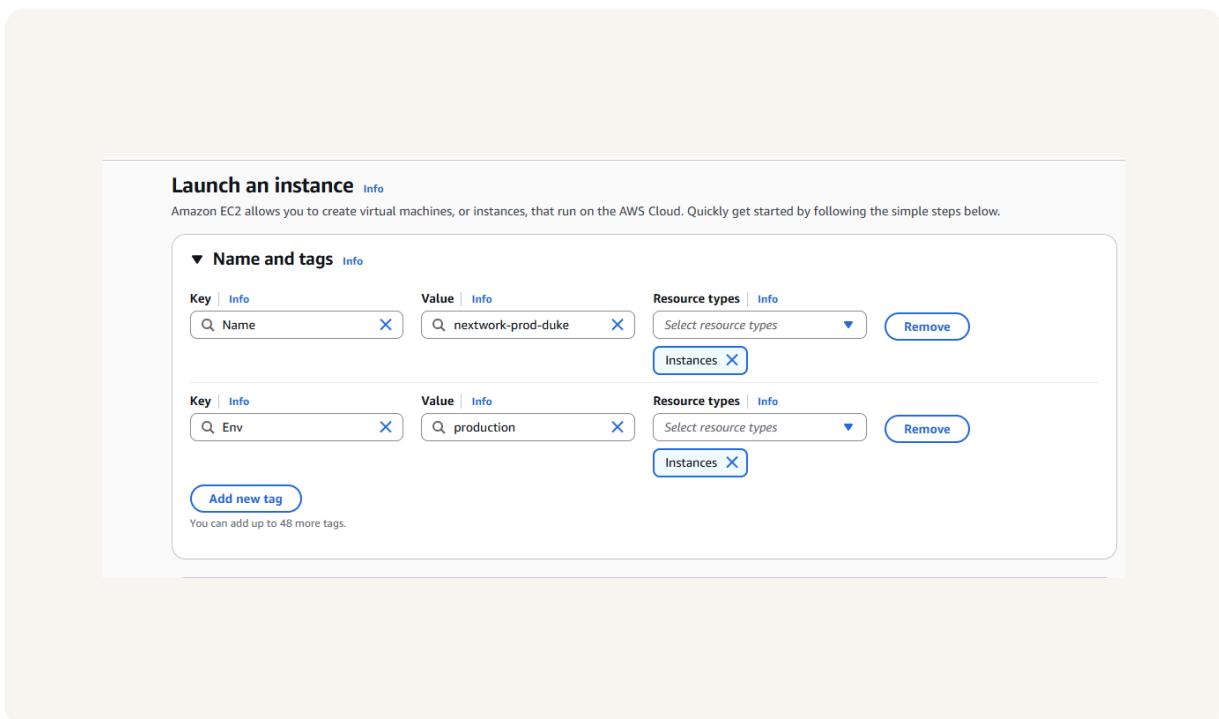
Project reflection

This project took me approximately 1 hours. The most challenging part was lauch EC2 instance, it need a subnet. It was most rewarding to create a subnet and know a little bit about it.

Tags

Tags are labels that you can attach to EC2 instance. identifying all resources with the same tag at once (they are useful filters when you're searching for something), cost allocation, and applying policies based on environment types.

The tag I've used on my EC2 instances is calle Name and Env. The value I've assigned for my instances are nextwork-prod-duke and production for production instacne and nextword-dev-duke and dev for development instance.



IAM Policies

IAM Policies are rules for who can do what with your AWS resources. It's all about giving permissions to IAM users, groups, or roles, saying what they can or can't do on certain resources, and when those rules kick in.

The policy I set up

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

I've created a policy that allows all actions that take on EC2 instance that has the Env - development tags. Allow the describe action and all ec2 instances and deny all the delete and create actions to all the ec2 instances.

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

Effect: whether the policy Allow(s) or Deny(s) the request. Action: which operations or API calls (e.g., ec2:StartInstances) the statement covers. Resource: which objects (ARNs or "*"for all) the rule applies to.

My JSON Policy

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

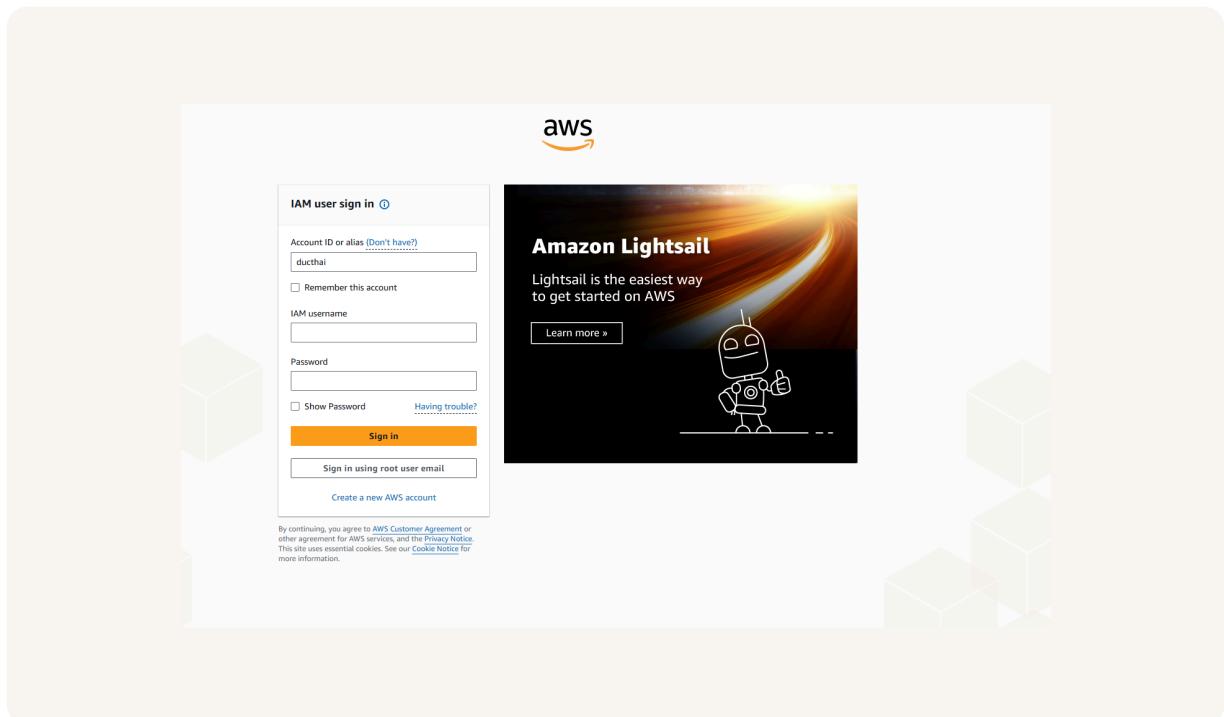
```
1 { "Version": "2012-10-17",  
2   "Statement": [  
3     { "Effect": "Allow",  
4       "Action": "ec2:Describe*",  
5       "Resource": "*"  
6     },  
7       "Condition": {  
8         "StringEquals": {  
9           "ec2:ResourceTag/Env": "development"  
10         }  
11       }  
12     },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

The right panel shows a sidebar titled 'Select a statement' with a button '+ Add new statement'.

Account Alias

An account alias is a friendly name for your AWS account that you can use instead of your account ID (which is usually a bunch of digits) to sign in to the AWS Management Console.

Creating an account alias took me one minute Now, my new AWS console sign-in URL is <https://ducthai.signin.aws.amazon.com/console>



IAM Users and User Groups

Users

IAM users are AWS identities for people or services. Each user can have login credentials and is granted permissions via policies or groups to control which AWS resources.

User Groups

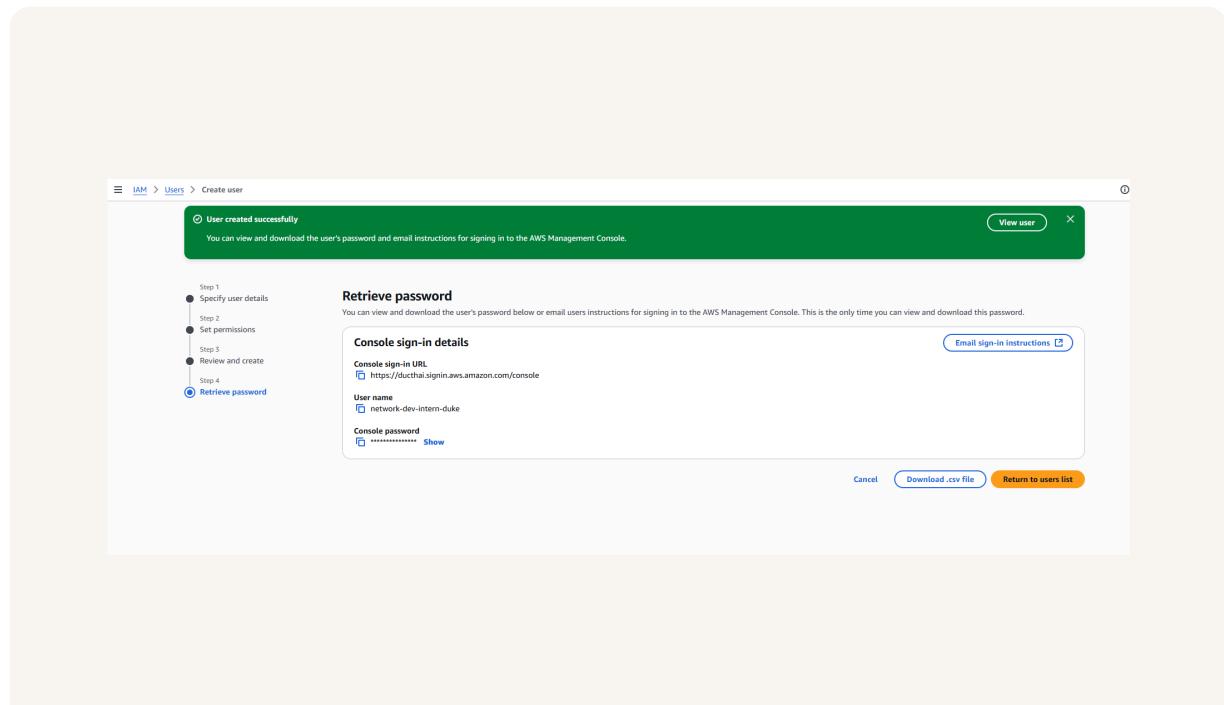
An IAM user group is a collection/folder of IAM users. It allows you to manage permissions for all the users in your group at the same time by attaching policies to the group rather than individual users.

I attached the policy I created to this user group, which means all user in the group have the similar access to AWS resources.

Logging in as an IAM User

The first way is thorough console with sign in url and password. Other way is programmatically through CLI/SDK use.

Once I logged in as my IAM user, I noticed my history, application, cost management is disabled. This was because AWS treat me as new user and I do not have access to all the resources.



Testing IAM Policies

I tested my JSON IAM policy by try stopping noth prod and dev instances. I failed to stop the prod instance but I was able to stop the development one.

Stopping the production instance

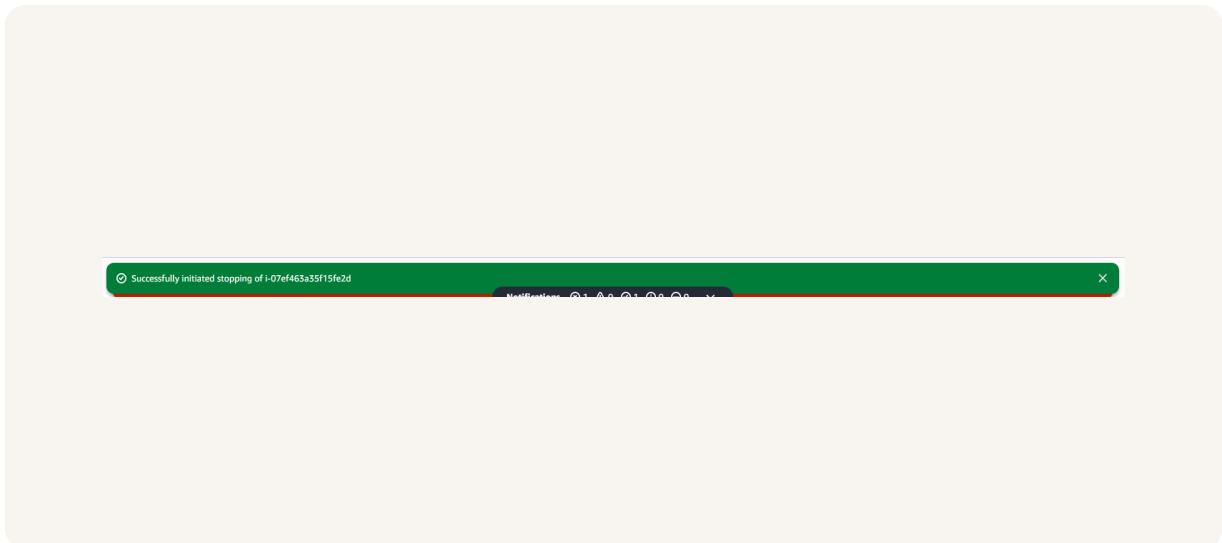
When I tried to stop the production instance an angry-looking banner tells us we've failed to stop this instance and I am not authorized. This was because my IAM policies does not allow me to stop any instance at all.

Failed to stop the instance i-0d92ba485b432ba27
You are not authorized to perform this operation. User: arn:aws:iam::841162690953:user/network-dev-intern-duke is not authorized to perform: ec2:StopInstances on resource: arn:aws:ec2:us-east-1:841162690953:instance/i-0d92ba485b432ba27 because no identity-based policy allows the ec2:StopInstances action. Encoded authorization failure message:
JH7_eMjf04mgv0Kqkv9_zlvUyN_ZRXdTmYLqN9FQYfb9uSsvK-DzZ68-ltvukxfmnH0sOlw5jTbz1Dv5_nKUsrBJN-
QKUrALGWJUpFts0qjk33sFr3uWE1LkhY5kxhaOaVYRCpYsj5PlCpMNv65g24IXd6WbODpuoi0lJ2JuNs_adn0H967BQpyBju1JLBmOowAudOuSpIheb_EBtiWk13cWvA1tjXeRyheQ6fhNjh4wkZ-
3zrWMxz2AD6viscqDkfUY2yrmqkQ33Vry_0ybt1Fbsr8Ry3faV9qUKGm8Nps6bxXW73KeH3lbaED01tGDQE39E52c9W8uqt-
x9yZT6U_ZL6UVggDH6Z_gKBKhVi4XVd7ASRCctpa35IEPdRjHrpVpqrhArbJy-
ulBjkJSiuE5OBqljgloh2EA1SE71a4Bin6DYODSlz5ke88f6f46YoCRBSEx7JNpNXMB5jESIIQH7XyZ5JgYLb0mBljz3Jkn2uhGHAS78hhMqIO-GKtJG9sp5DscCrdOfZ1nvbxyl-
3QH6IdOG8yo0SVhmlmDlh4cPb3IM-k08jQplAPPTjBy_3GstTpVfgeyhbMT9_saNu2X8UwDLXKvzReEzRvdQj4yXv65QvAOYk-
CdAuglbs4kmb5j4rvZPNsyijNB9shJnNqz5FAKB4703sMLE_OfrKg1HxnAq8lEDad_pWq0g1gbmT6Z1cpcquCUjsAeWCtr4iBjOIP3Uy4lr-7bRYlrOmsT-
ESM1PYloqGpxloGbPScT_PlrebXoOx3sVhZoQR9G3PNwsZ5y1a00Gv5lqpXOmMuCAmFNmxJ3LjML466Y0mMyhY4gfUp5s6GzzCvoUqY50AbQ

Testing IAM Policies

Stopping the development instance

Next, when I tried to stop the development instance I was sucessful. This was because the IAM pocilies what attched to me allows me to any action to development instance.





nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects

