General Immersion Day

Lab 4

IAM Hands on Lab

Identity & Access Management (IAM) Overview

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is **authenticated** (signed in) and **authorized** (has permissions) to use resources.

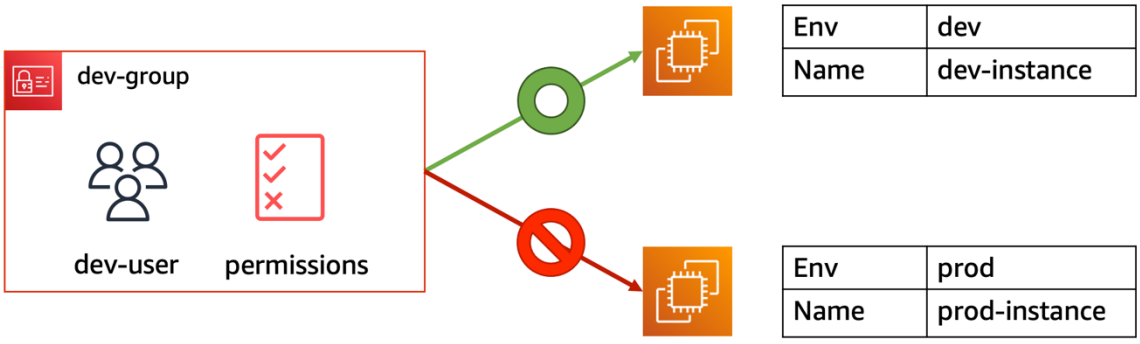
When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account ***root user*** and is accessed by signing in with the email address and password that you used to create the account.

Note: We strongly recommend that you **do not use the root user** for your everyday tasks, even the administrative ones. Instead, create and use **IAM user** by referring to the instructions here .

AWS IAM key components:

- IAM Identities
  - **IAM Users**
  - **IAM User groups**
  - **IAM Roles**
- **IAM Policy**

And to help secure your AWS resources, check the document for security best practices in IAM .



This hands on lab is broken into the following parts:

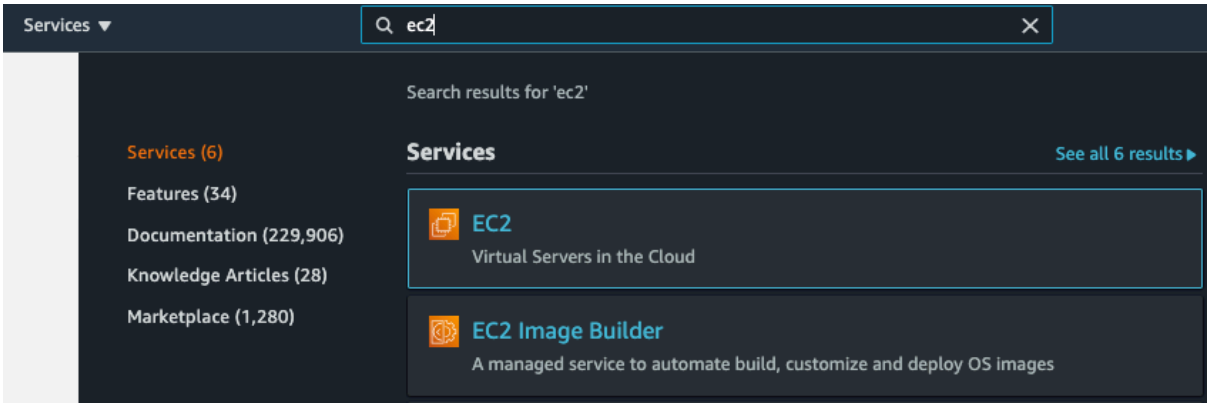
1. Launch EC2 Instances with Tags
2. Create AWS IAM Identities
3. Test the access for resources
4. Assign IAM Role for EC2 Instance and Test the access

4-1 Launch EC2 Instances with Tags

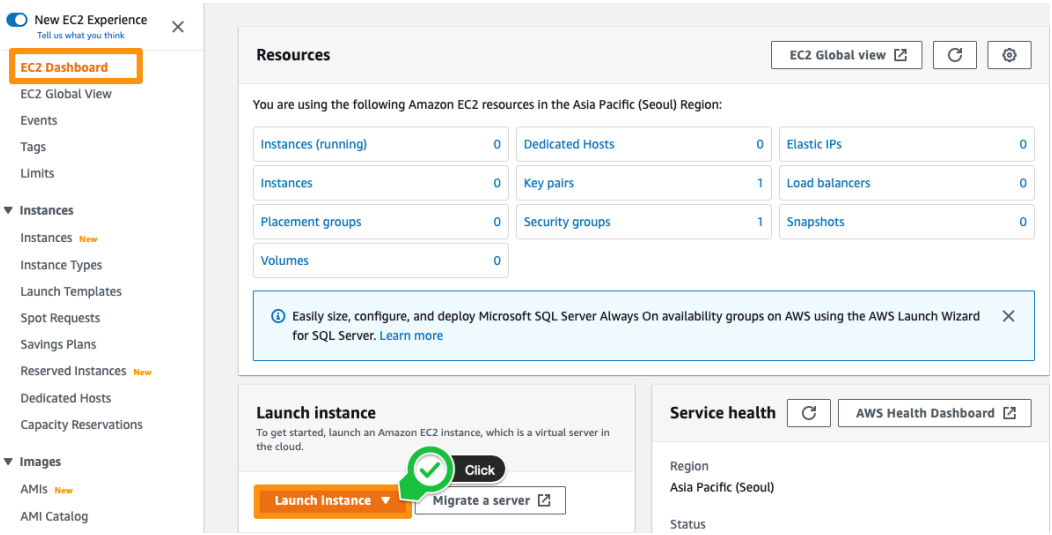
In this lab, we will launch two Amazon Linux 2 instances. Suppose one is an EC2 instance used in a development environment and the other one used in a production environment. We will use **tags** to distinguish these two instances.

---

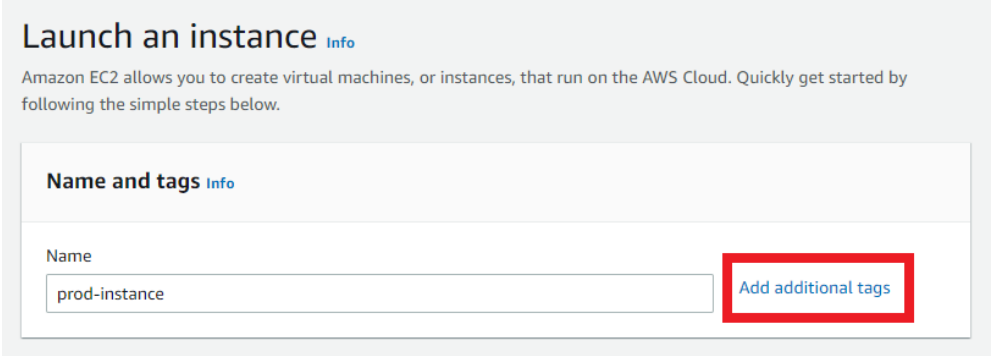
1. Enter into EC2 Console in AWS Management Console.



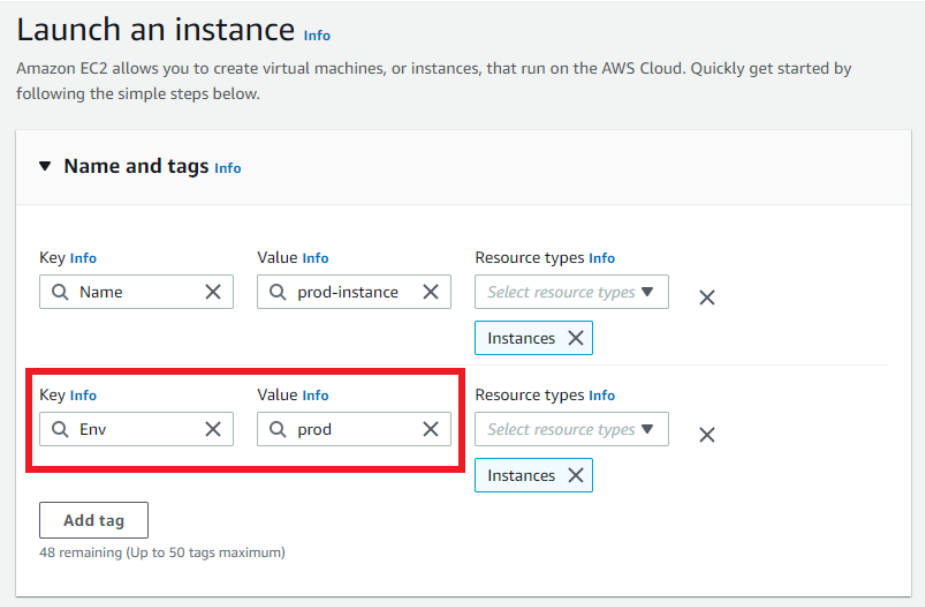
Click on **EC2 Dashboard** near the top of the leftmost menu. And Click on **Launch instances**.



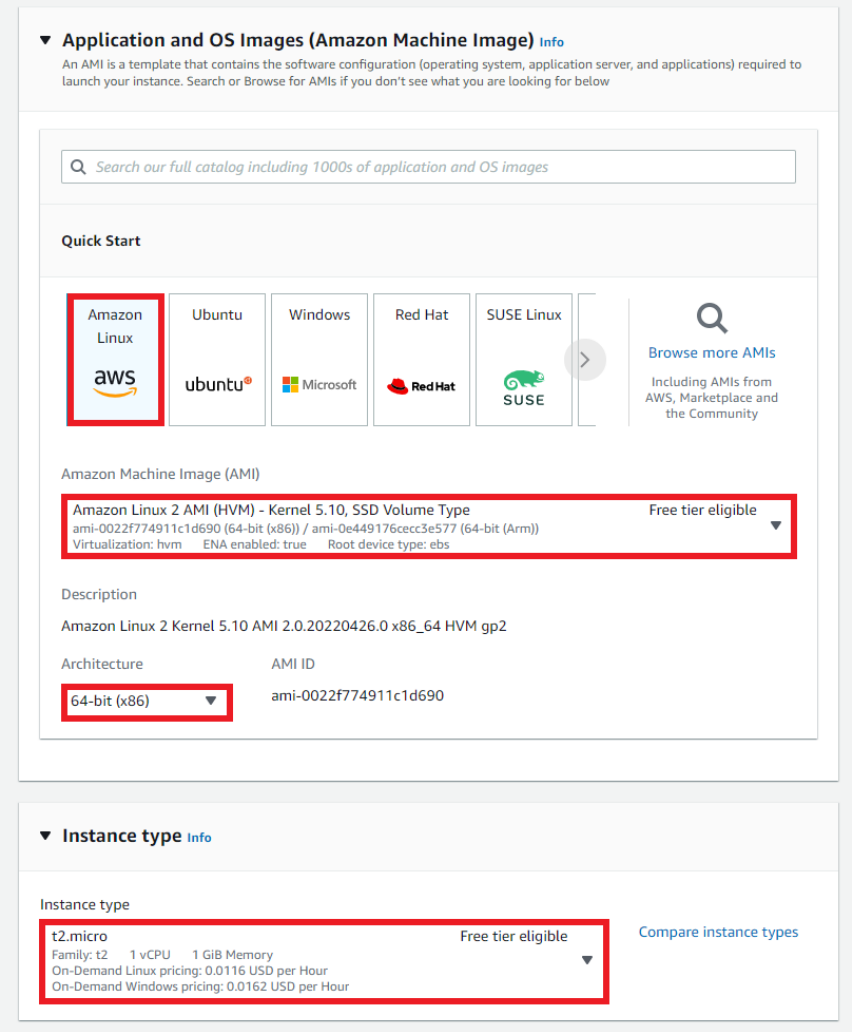
2. In **Name**, enter the value **prod-instance**. And click on **Add additional tags**.



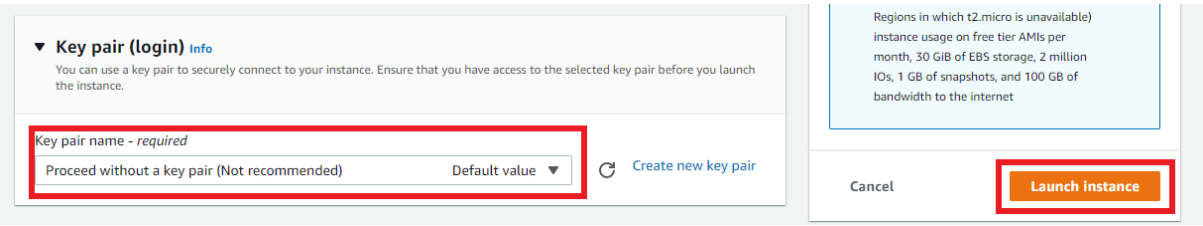
3. Click **Add tag** then in **Key** enter **Env**, and **Value** enter **prod**.



4. Check the default settings for Amazon Machine Image and select **t2.micro** in Instance Type.

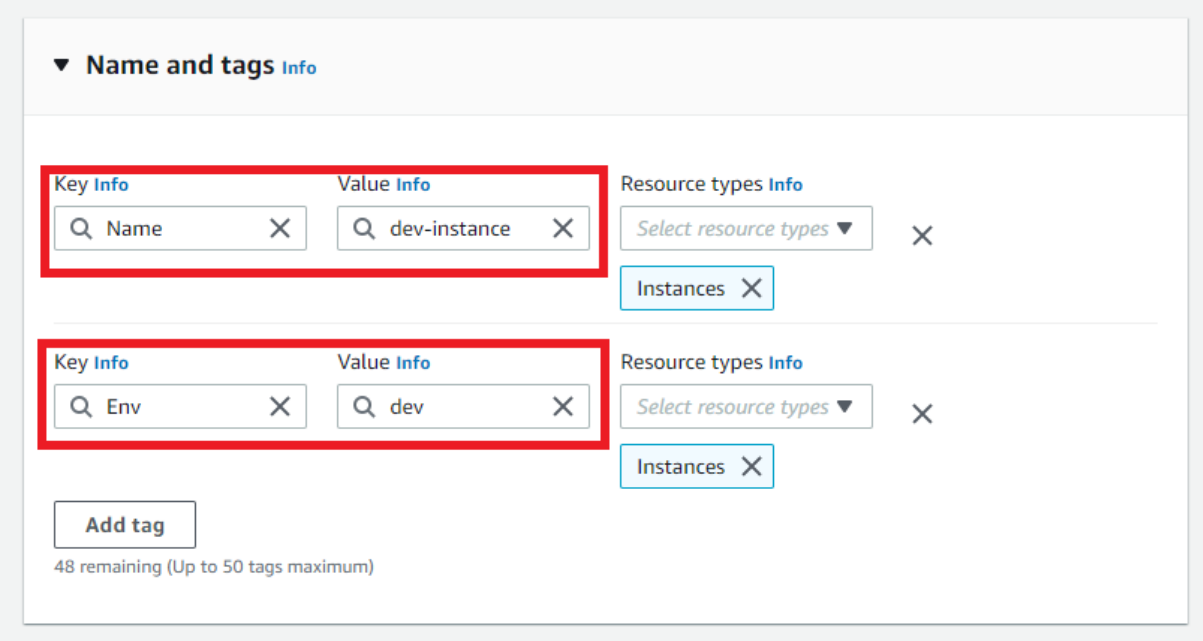


5. In **Key pair (login)** select **Proceed without a key pair**. Then click **Launch instance**.

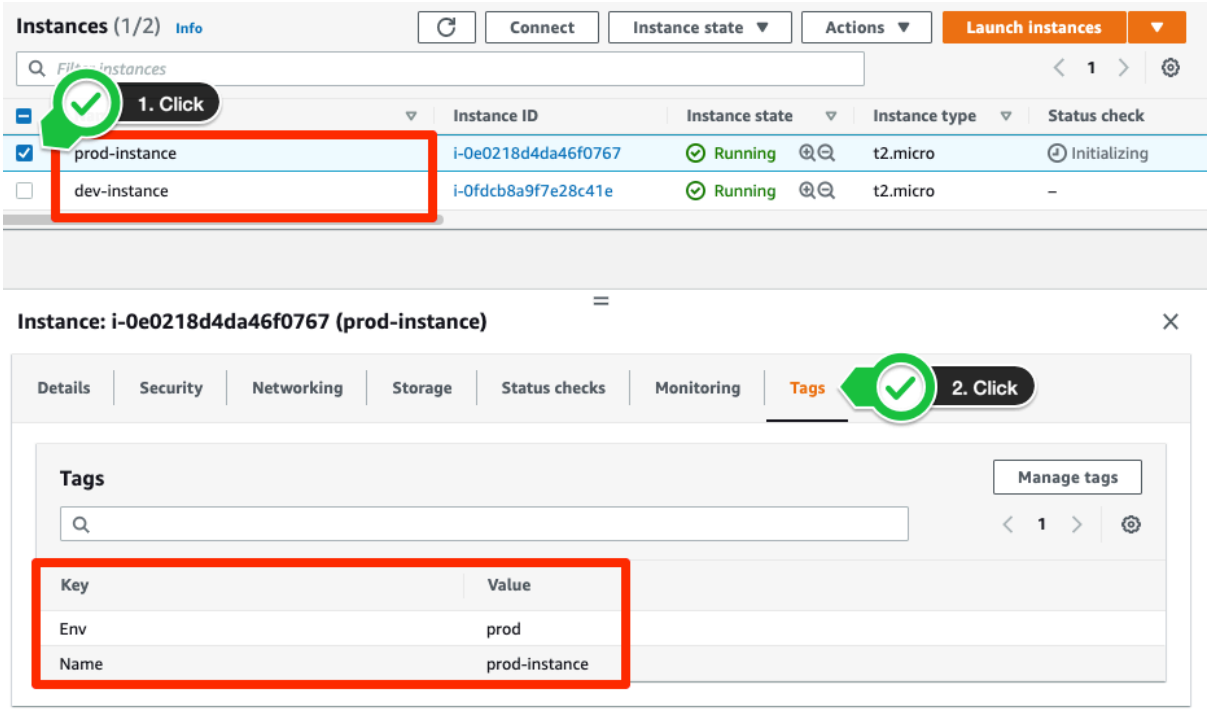


6. You have launched an EC2 instance for production environment. Now you have to create **one more EC2 instance for development environment**. *Repeat the above instructions(start with no.1 to no.5)*, but with a different Name and Env tag in step no.2 and step no.3 as follows.

KEY	VALUE
Name	dev-instance
Env	dev



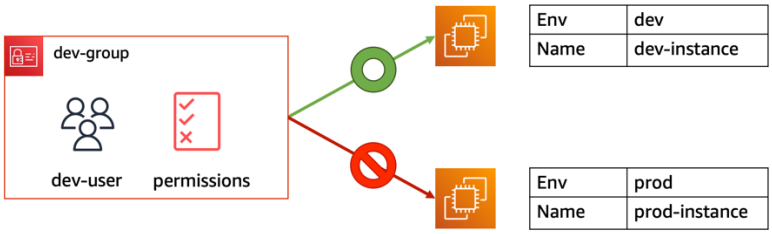
7. After you launched two instances, you can find them in the instances menu in the sidebar. Click the prod-instance checkbox and click the **Tags** tab in the bottom of the page. You can see details about this EC2 instance's tag information.



Great Job! You have deployed two instances(servers) with different resource tags.

4-2 Create AWS IAM Identities

In this chapter, we will work on creating **AWS IAM Identities**. AWS IAM Identity includes IAM Users, IAM User groups, and IAM Roles. Also, we will work on creating **IAM Policy** an object in AWS that, when associated with an identity or resource, defines their permissions.

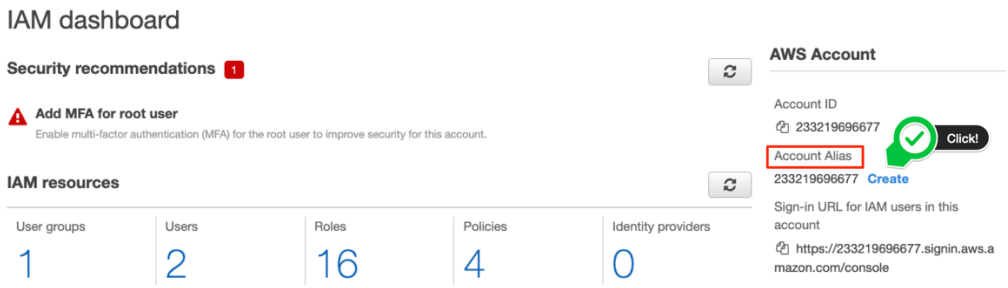


This chapter consists of the following steps.

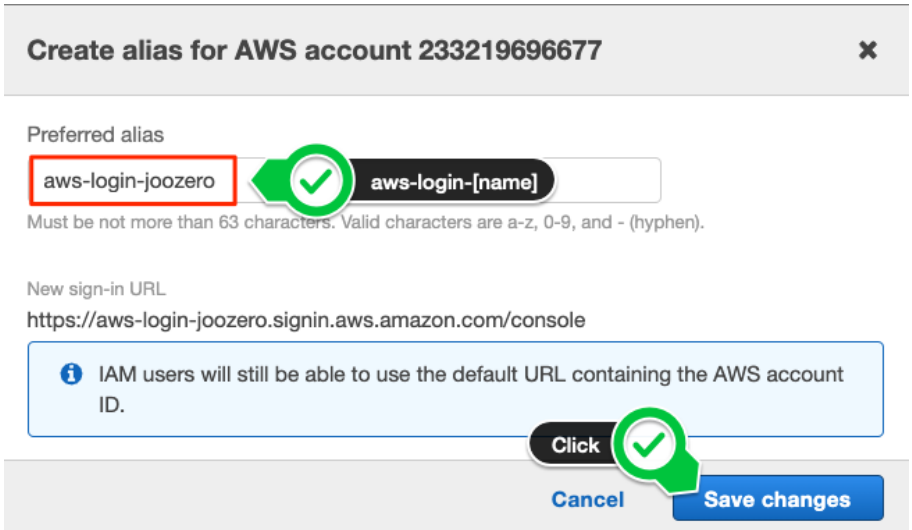
- Create **IAM Policy** to attach to IAM User group
- Create **IAM User group** named dev-group
- Create **IAM User** which name is dev-user placed in dev-group

---

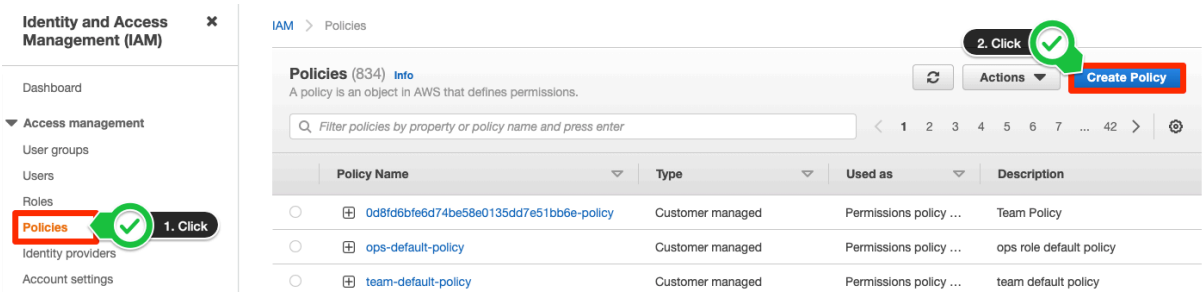
1. Log in to IAM console . To generate a console login **Sign-in URL**, click the **customize** button as shown below.



2. Input account alias. For this lab, type aws-login-user_name and click **create alias** button.



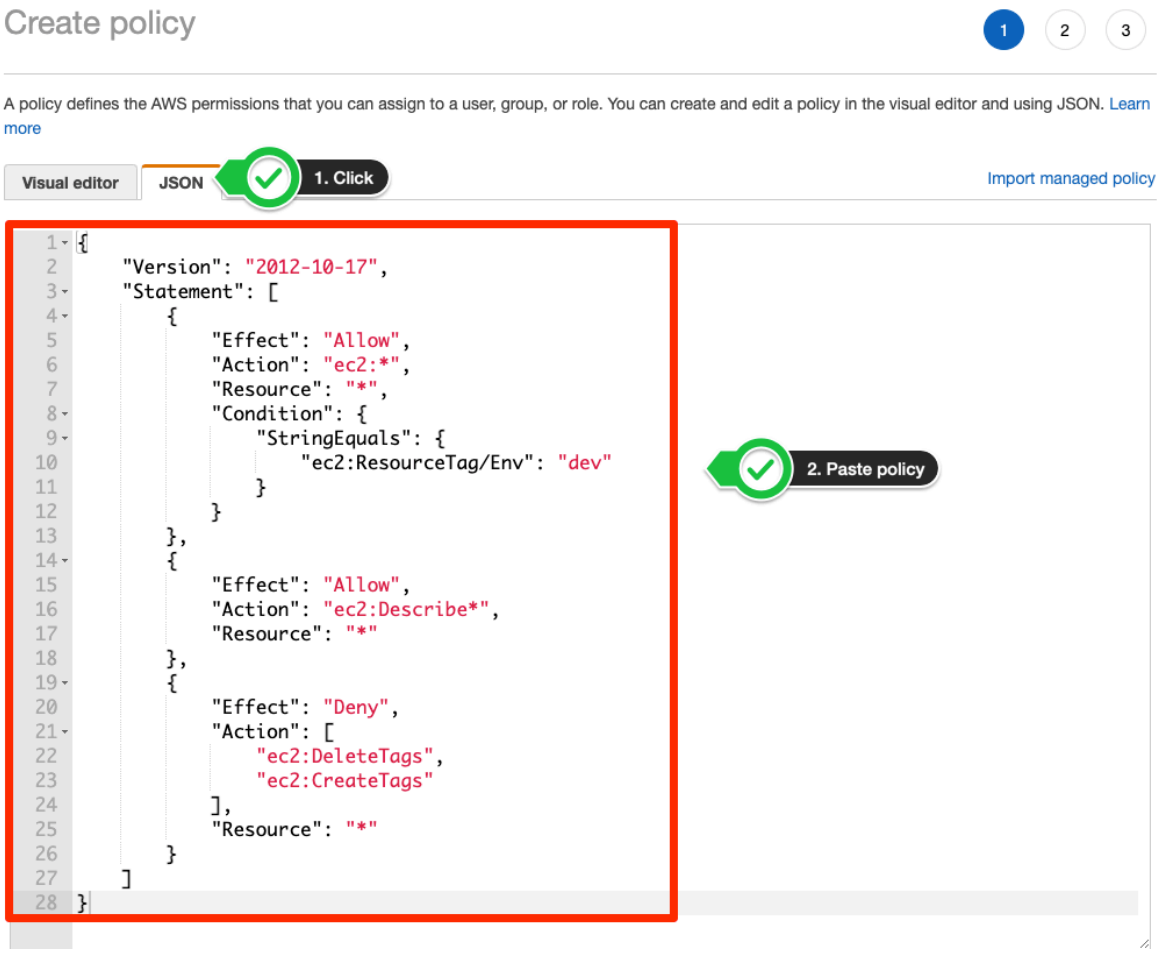
3. Click **Policies** on the left side of the IAM console, and then click the **Create Policy** button placed at the top of the right corner.



4. When defining policy for the AWS permissions, you can create and edit in visual editor or JSON. In this lab, we will use **JSON** method. Briefly describe the permission below, this policy allows all actions for **EC2 tagged as** Env-dev. Also, it **allows describe-related actions** for all EC2 instances. But it **denies create and delete tags action** to prevent users from modifying tags arbitrarily. Note that deny effect takes precedence over allow effect.

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

Click the JSON tab and paste the above policy and click **Next: Tags** button.

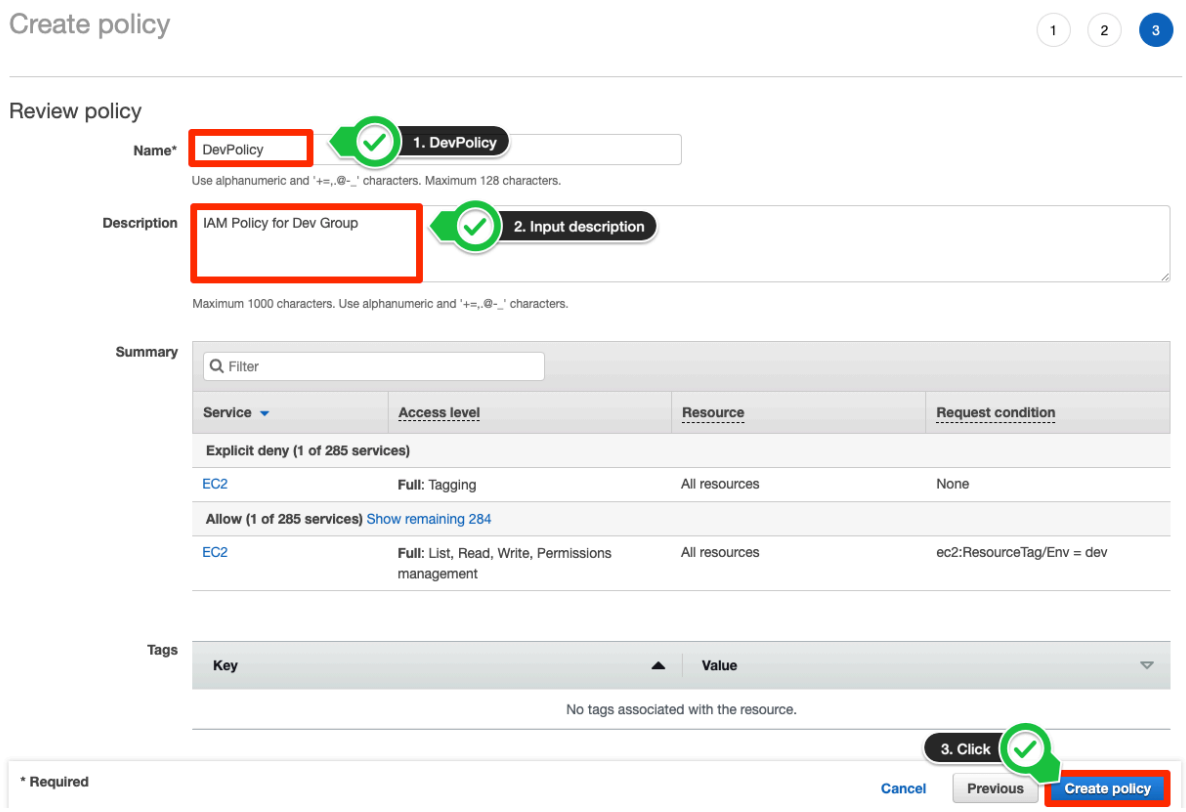


If you want, you can read more on [JSON policy structure]

- Version: use the latest 2012-10-17 version
- Statement: can include more than one statement in a policy
- Sid(optional): an optional statement ID
- Effect: use **Allow** or **Deny** to indicate the policy allows or denies access. **Deny** has priority
- Principal: If you create a resource-based policy, you must indicate the account, user, role, or federated user to which you would like to allow or deny access. If you are creating an IAM permissions policy to attach to a user or role, you cannot include this element. The principal is implied as that user or role.
- Action: a list of actions that the policy allows or denies
- Resource: If you create an IAM permissions policy, you must specify a list of resources to which the actions apply. If you create a resource-based policy, this element is optional. If you do not include this element, then the resource to which the action applies is the resource to which the policy is attached.
- Condition Block(optional): Specify the circumstances under which the policy grants permission

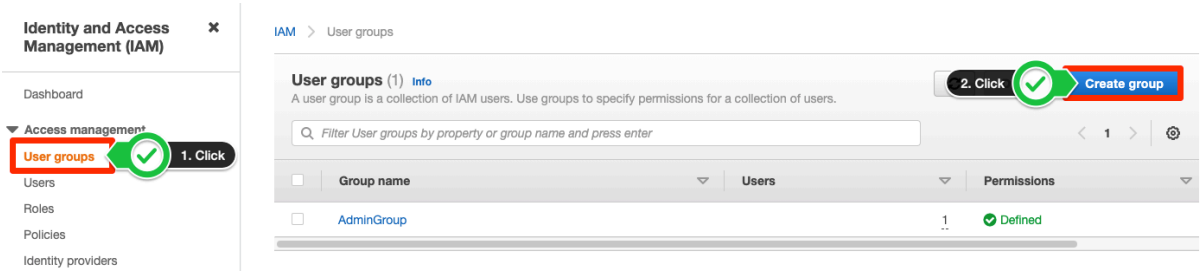
---

5. Keep default settings on next step, click **Next: Review** button. Write down DevPolicy in the name section, and write down description for this policy. Next click the Create policy button.

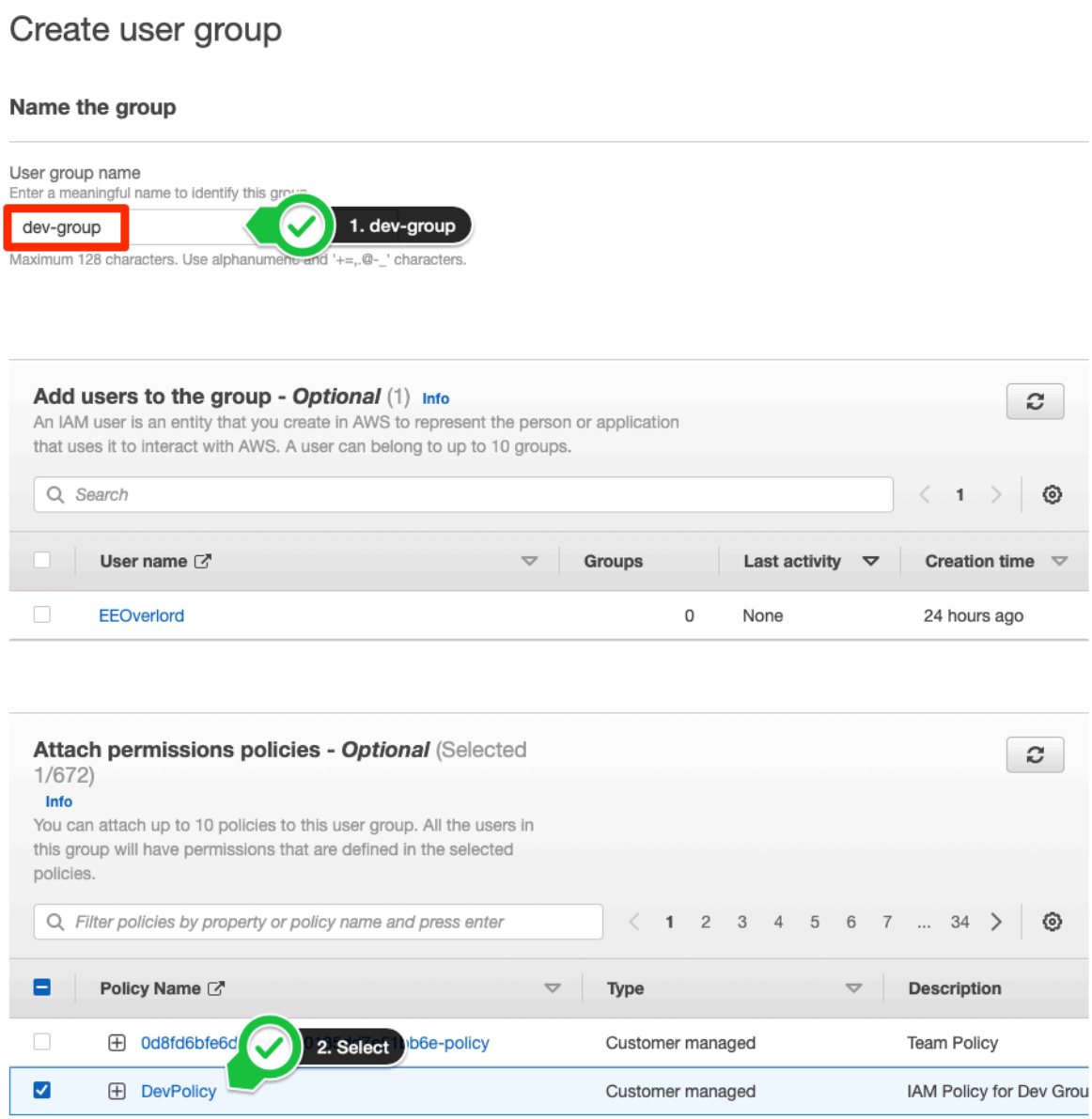


KEY	VALUE
Name	DevPolicy
Description	IAM Policy for Dev Group

6. Click **User groups** on the left side of the IAM console, and then click the **Create group** button placed at the top of the right corner.

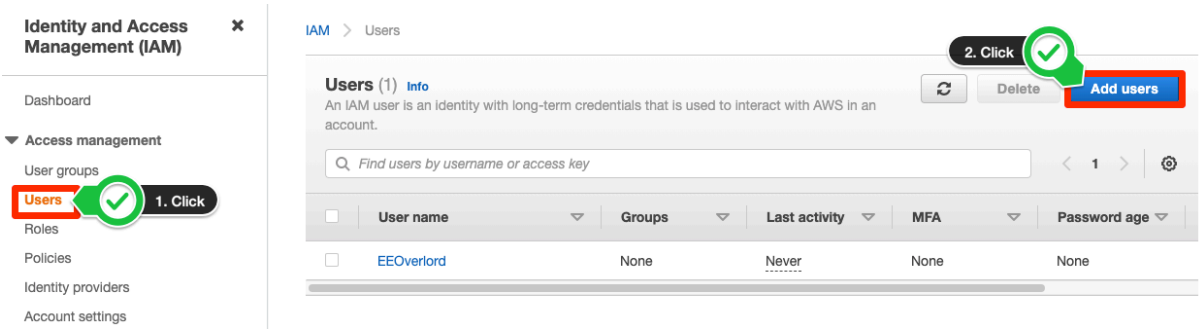


7. Type dev-group for **User group name** and select DevPolicy in **Attach permissions policies - Optional** section.

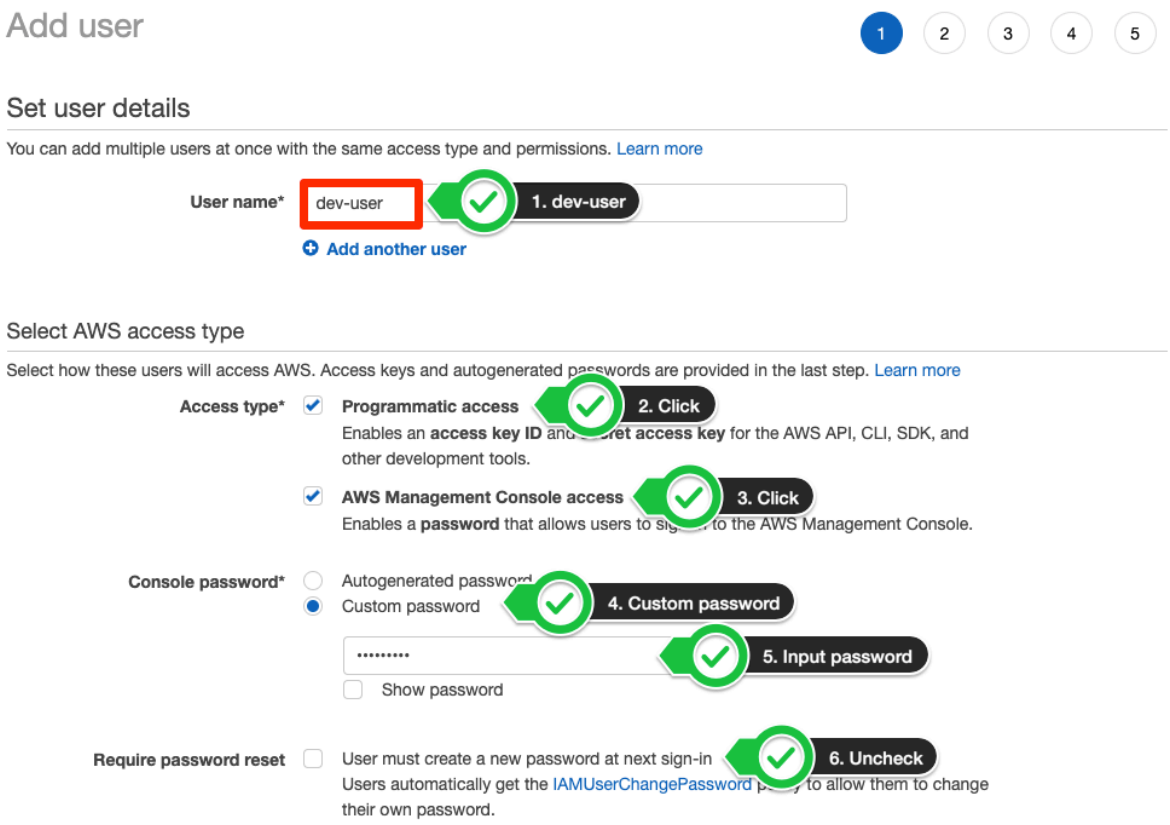


KEY	VALUE
User group name	dev-group

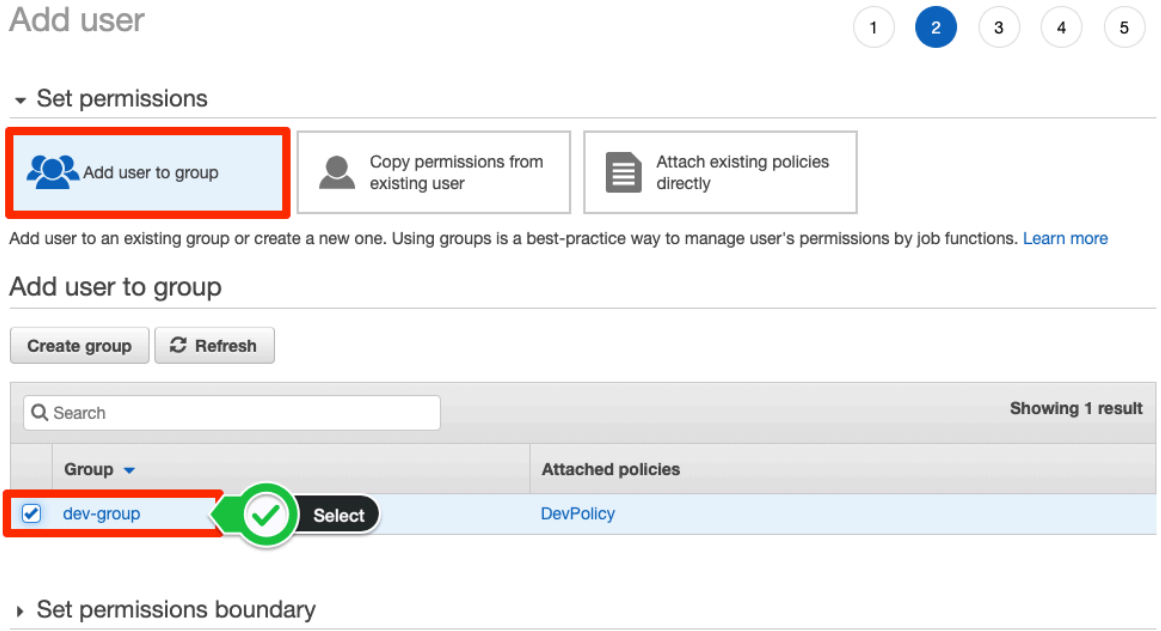
8. Click **Users** on the left side of the page, and then click the **Add users** button.



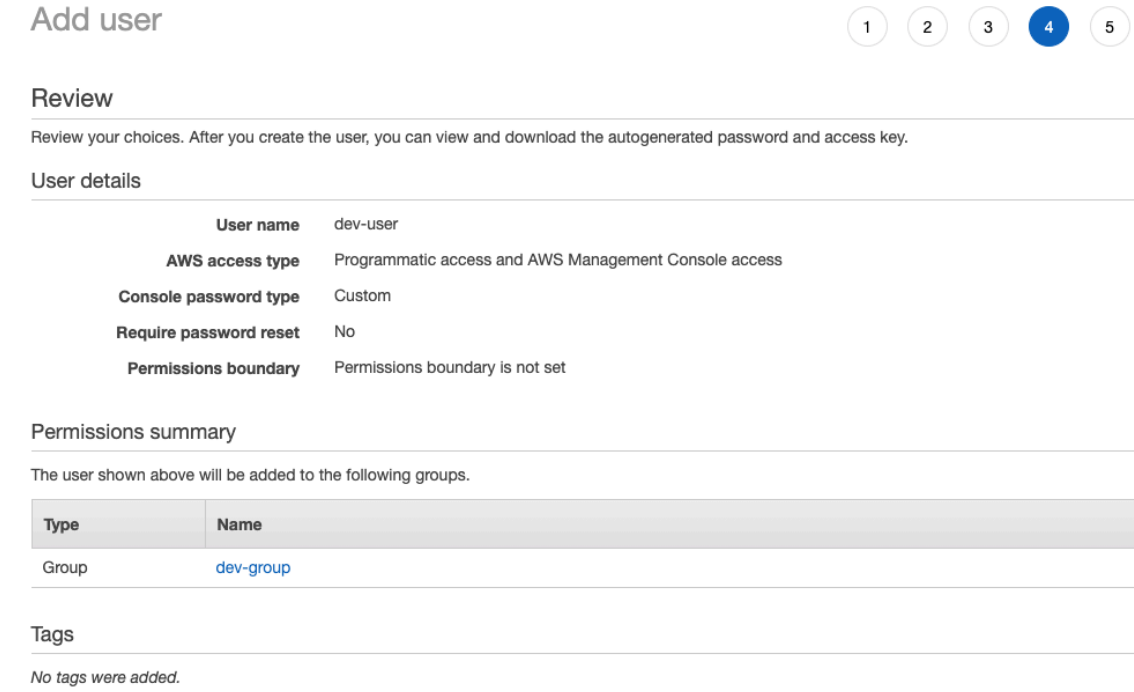
9. Type dev-user on **User name** and allow two access type both Programmatic access and AWS Management console access. And then, select **Custom password** and type password that you want. Finally, uncheck Require password reset function for quick action. In real world, it is recommended to activate the password reset. Click **Next: Permissions** button.



10. Select dev-group that we created right before and click **Next: Tags** button. Skip Add user page and move on to next step.

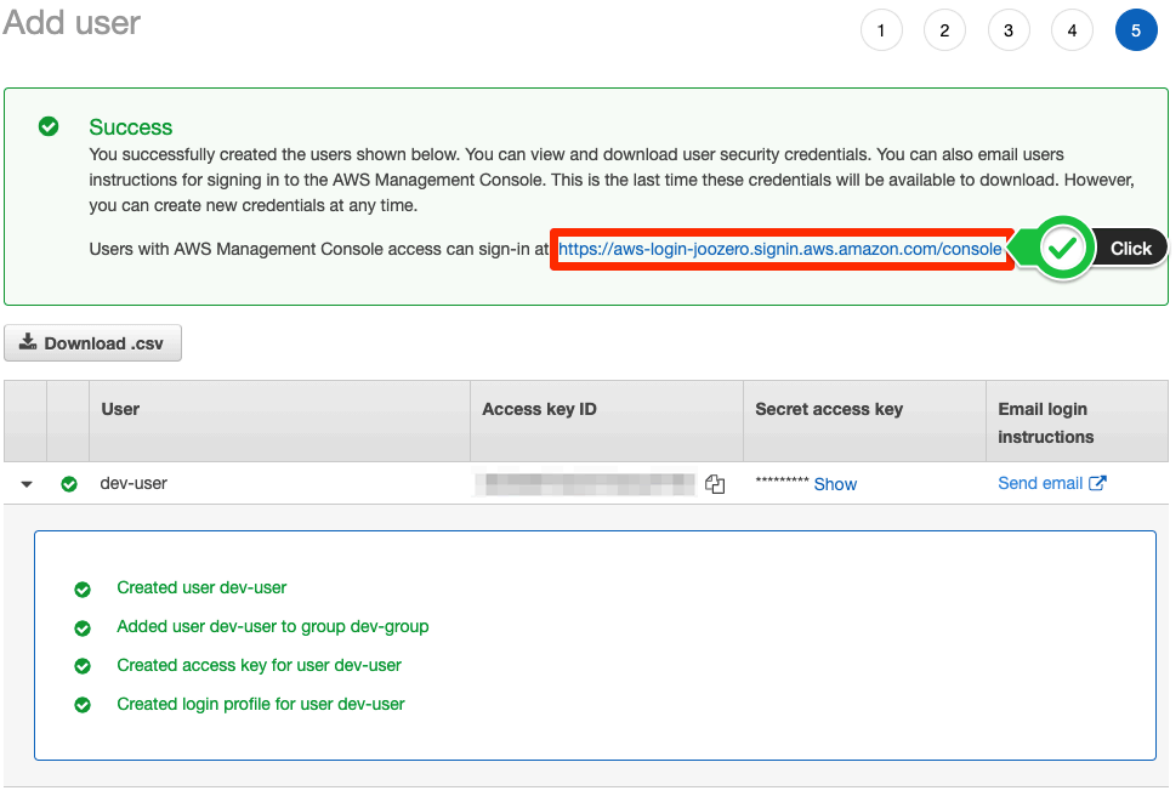


Click **Create user** button for adding dev-user.

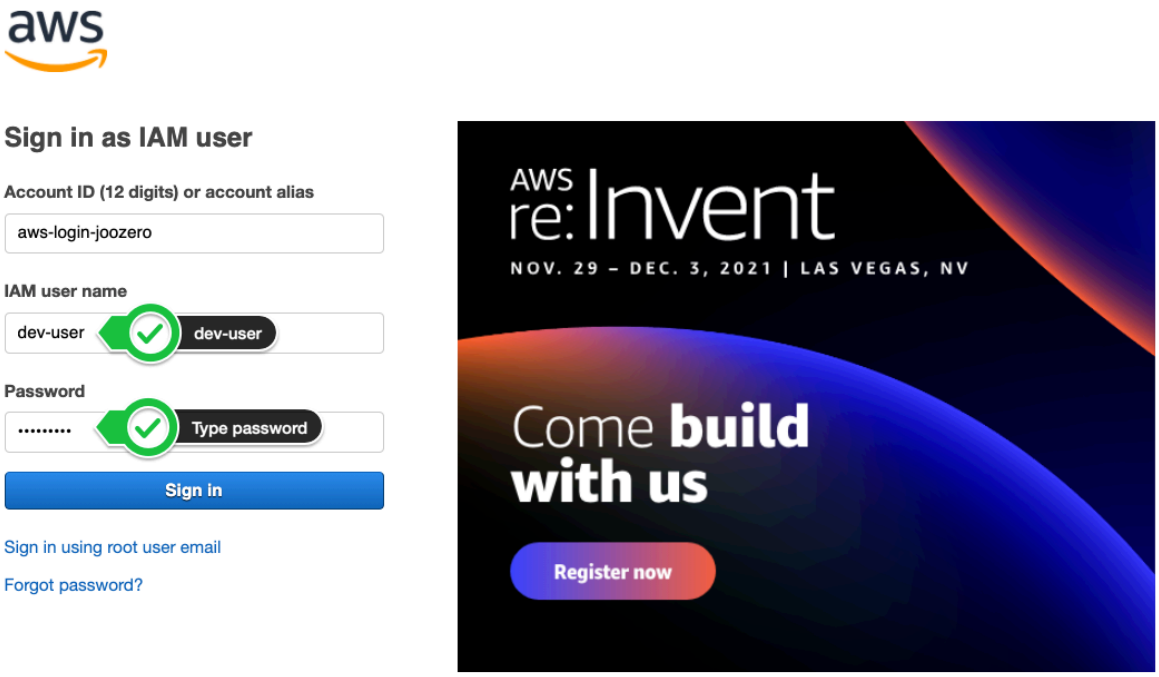


Download csv file to attain Access key ID and Secret access key. Also, you can send instructions for login.

11. Click **Sign-in URL** to log in as dev-user.



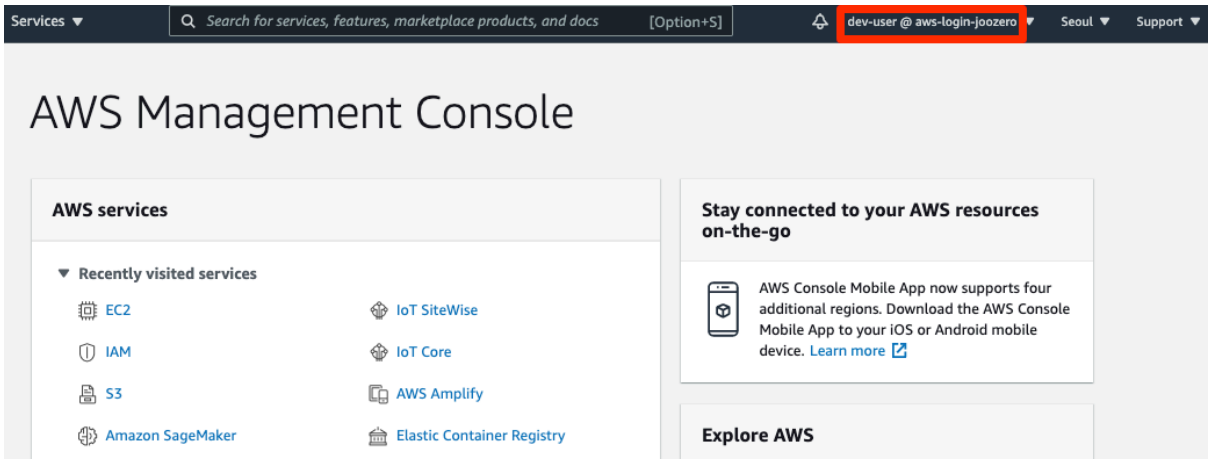
12. Type **IAM user name** and **Password** for log in and sign in for entering into AWS management console.



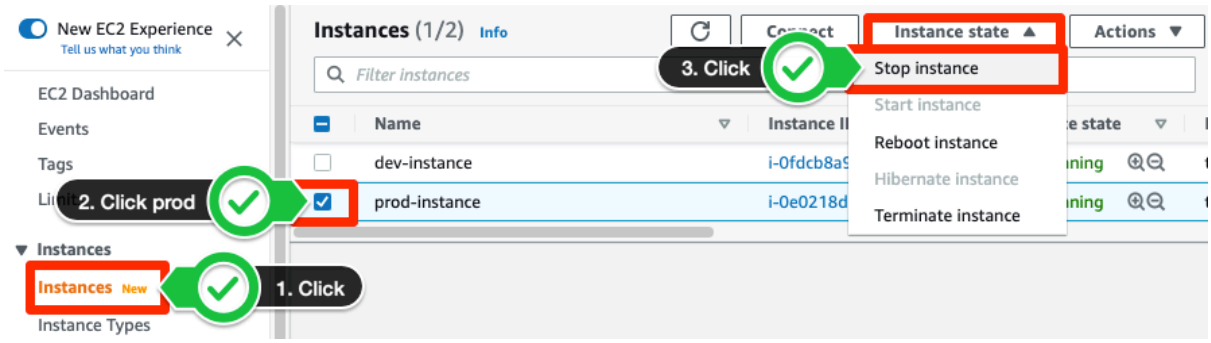
Great Job! You have created AWS Identities. Next chapter, we will check whether the permission we set works properly.

4-3 Test the access for resources

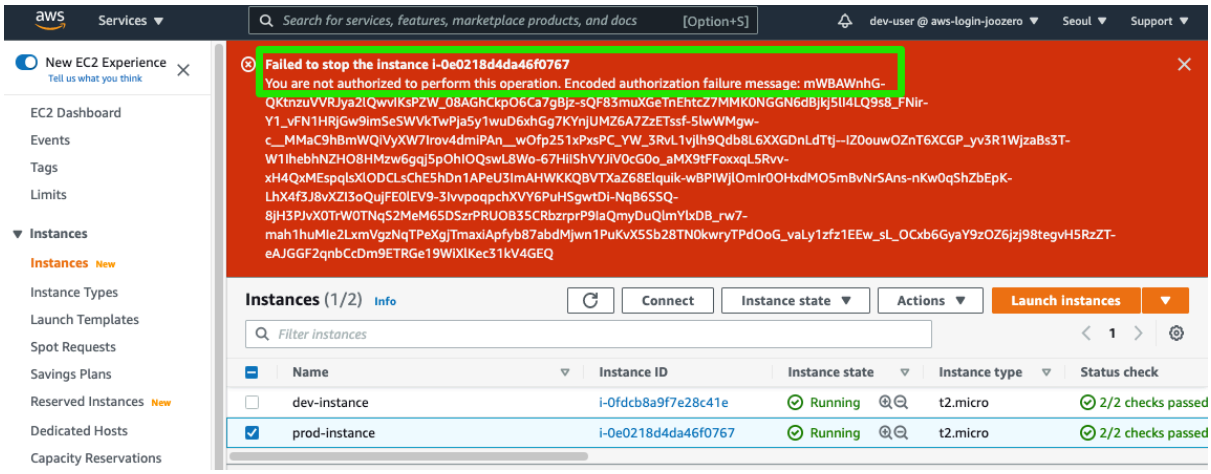
1. Log in AWS console and check **IAM User** and account alias. Also, check **AWS region** that launched EC2 instances are located.



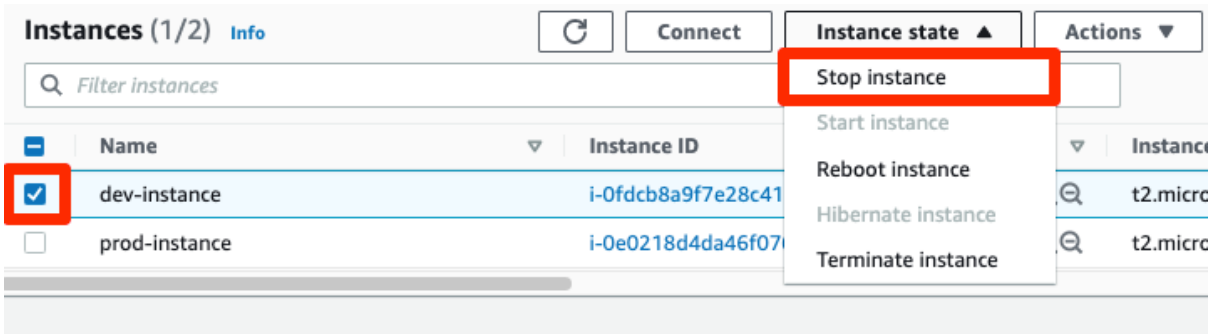
2. Move on to the [EC2 console] and click **Instances** menu. Select the instance named prod-instance and click **Instance state** button and **Stop instance** button. And click **Stop** button in the pop-up window.



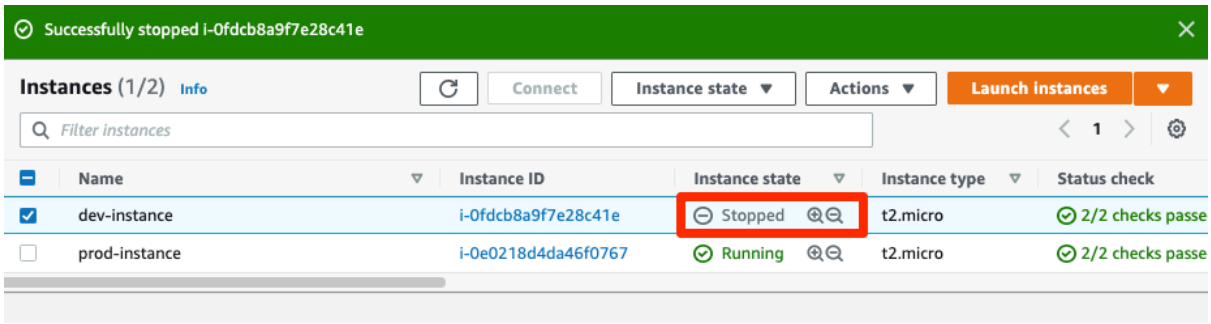
3. A warning sign appears that dev-user doesn't have permissions to perform the stop EC2 instance operation. This is because the previous hands on lab only allowed **dev-user** to perform the stop action on the ec2 instances with the resource tag **Env(key)-dev(value)**.



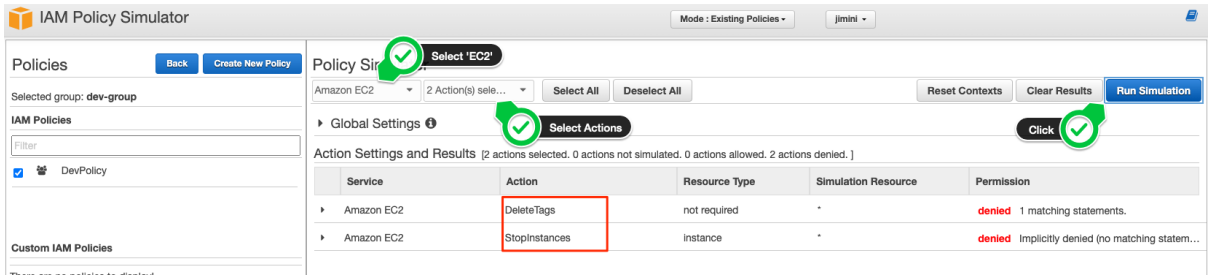
4. Select the instance named dev-instance and click **Instance state** button and **Stop instance** button. And click **Stop** button in the pop-up window.



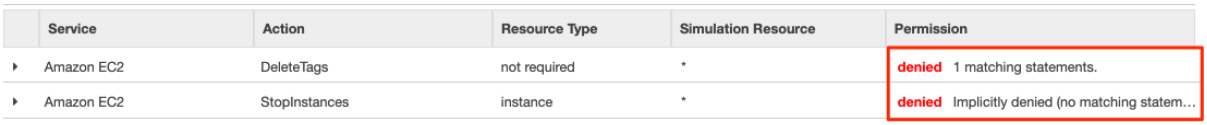
5. After a few seconds, you can see the dev instance has been stopped.

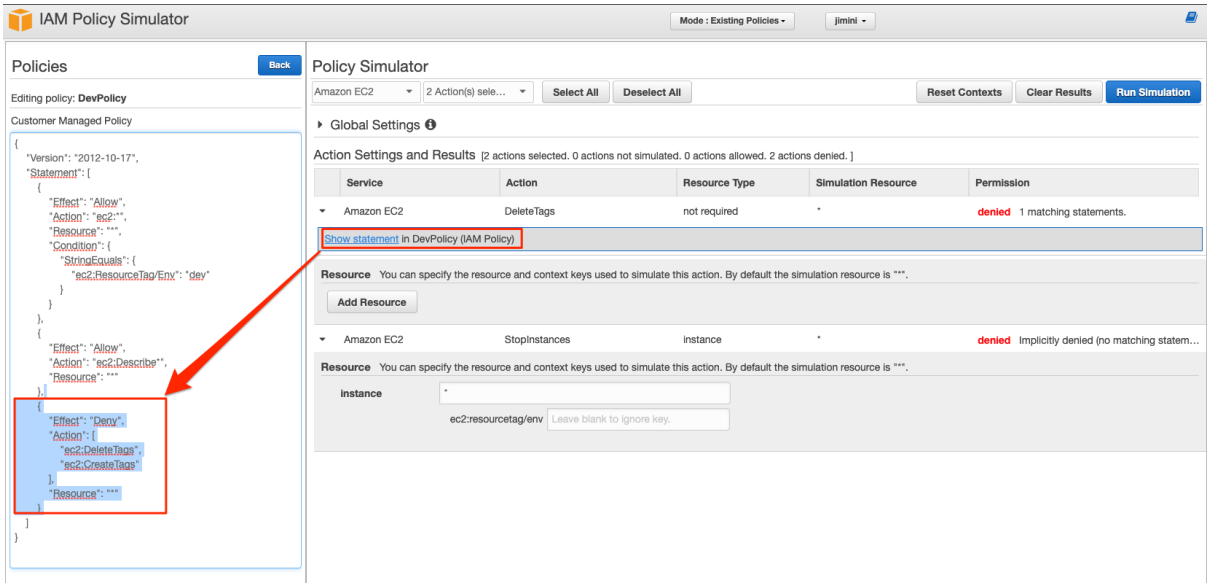


Great Job! You tested the permission for each EC2 instance. Next chapter, we will learn about AWS IAM Role for AWS resources!

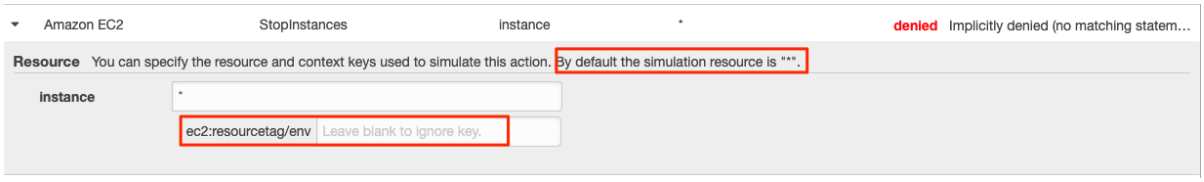


4. You can see both actions are denied but with different reasons. **DeleteTags** was denied because of **1 matching statements** and **StopInstances** was **Implicitly denied (no matching statements)**.

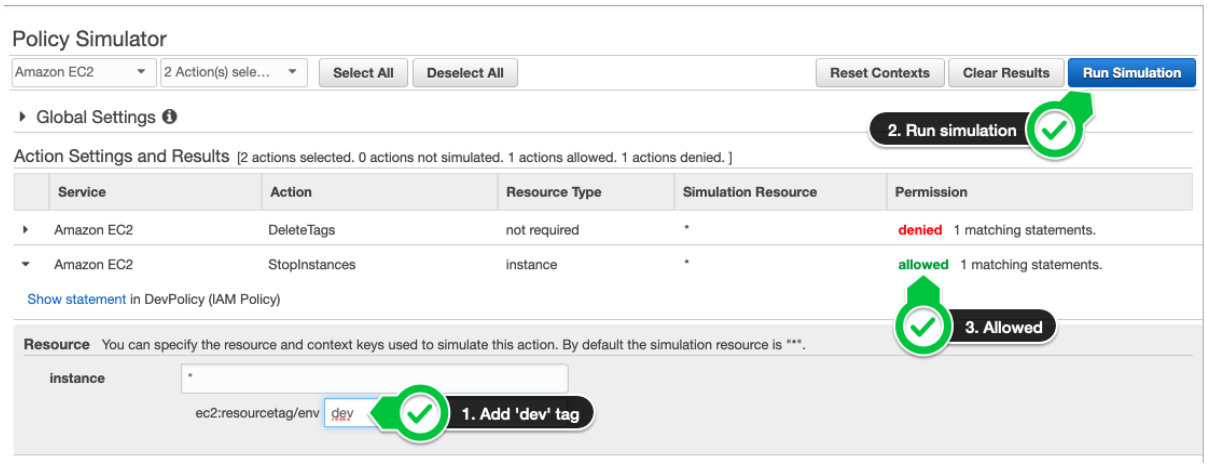




6. Expand StopInstances. The action was denied *because the simulation resource is* **"*"**. Please note that **dev-group** can only stop EC2 instances having **dev** tags.



7. Now we will test the policy after adding **dev** tag to validate the policy correctly.

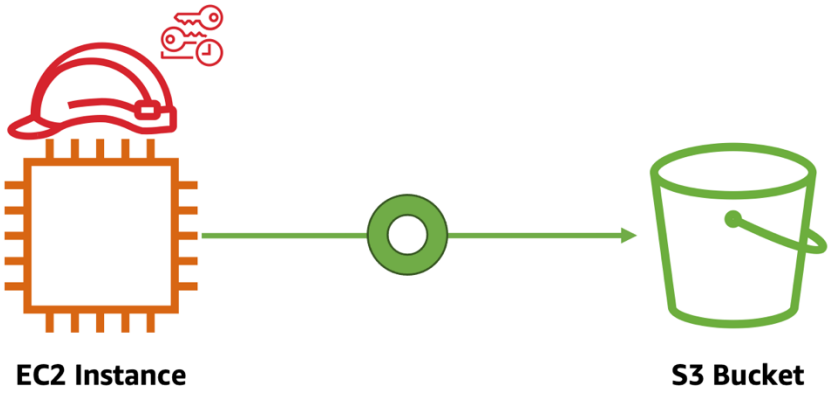


Great Job! You just tested the your custom IAM policy without affecting your environment.

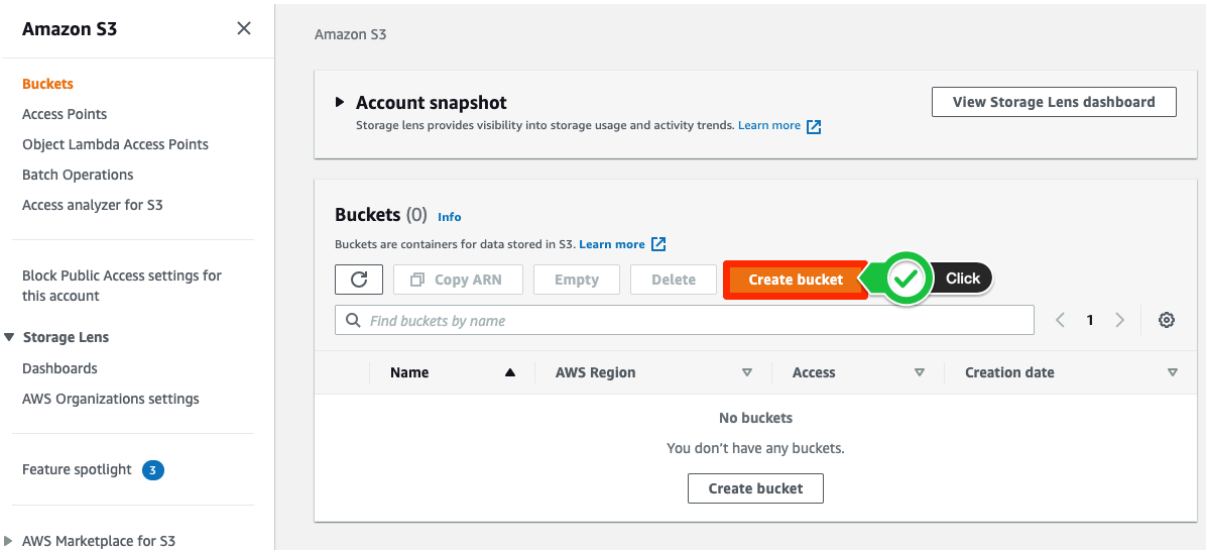
With the IAM policy simulator, you can test and troubleshoot identity-based policies, IAM permissions boundaries, Organizations service control policies (SCPs), and resource-based policies. You can learn more details here.

4-4 Assign IAM Role for EC2 Instance and Test the access

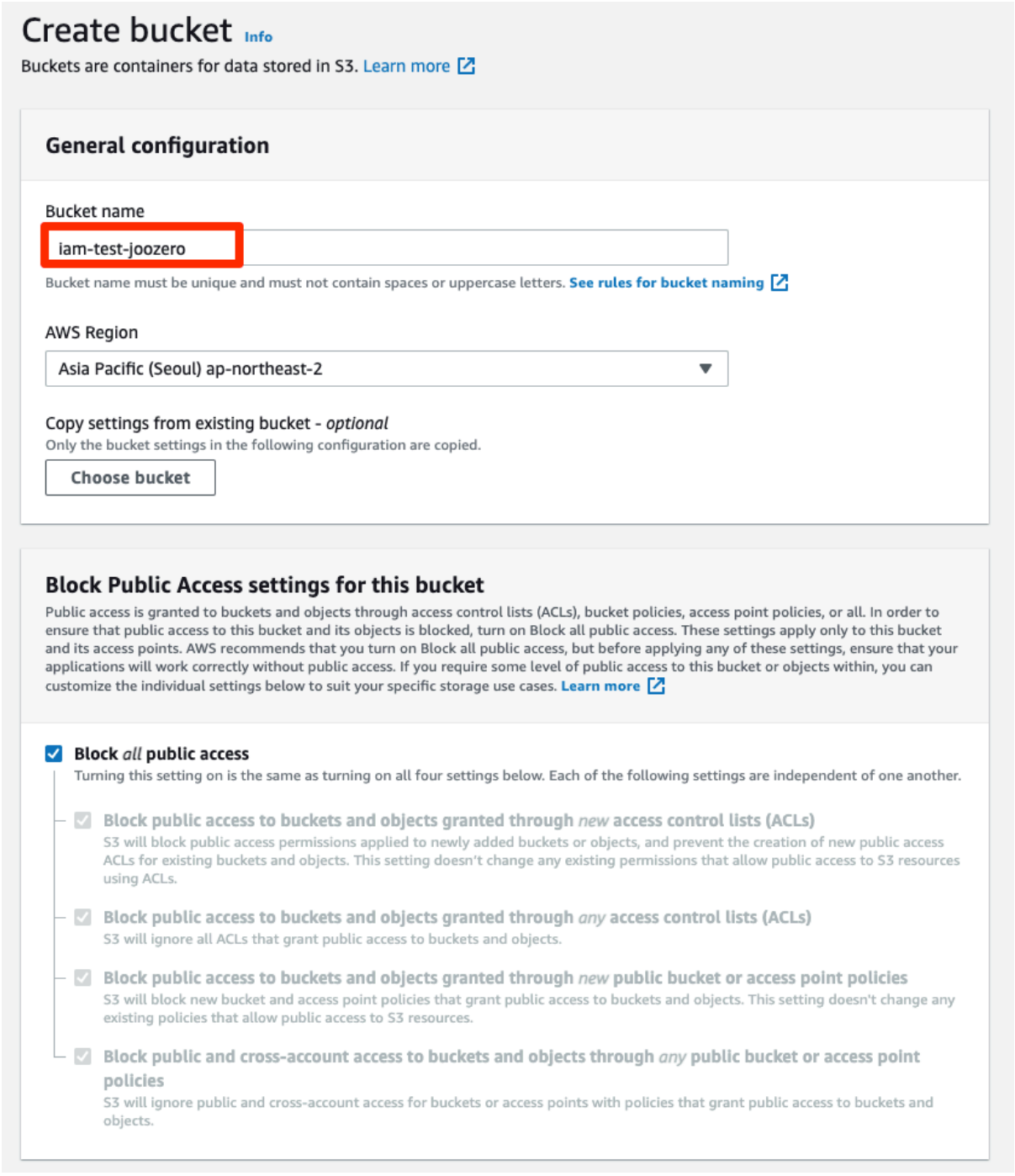
Re-login to the **AWS account with administrator role** at the beginning of the hands on lab, not dev-user before proceeding with this chapter.



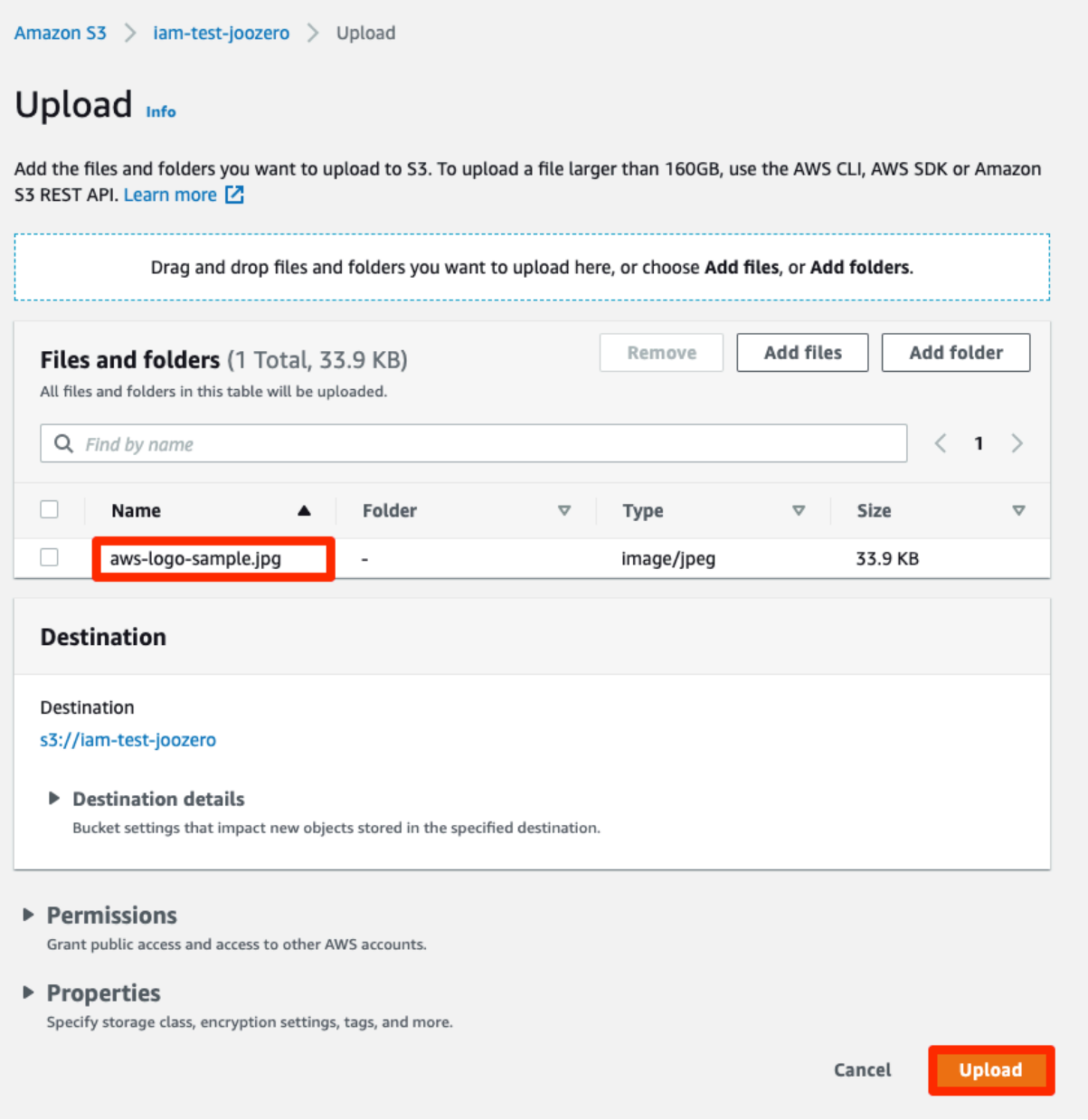
1. To create S3 Bucket, enter into the [S3 console](). And then, click **Create bucket** button. For more detail information on Amazon S3, please refer to the Storage on AWS chapter.



2. Enter a unique bucket name in the **Bucket name** field. In this lab, type iam-test-user_name. All bucket names in Amazon S3 have to be unique and cannot be duplicated. Click Create bucket button, leaving the remaining settings intact.

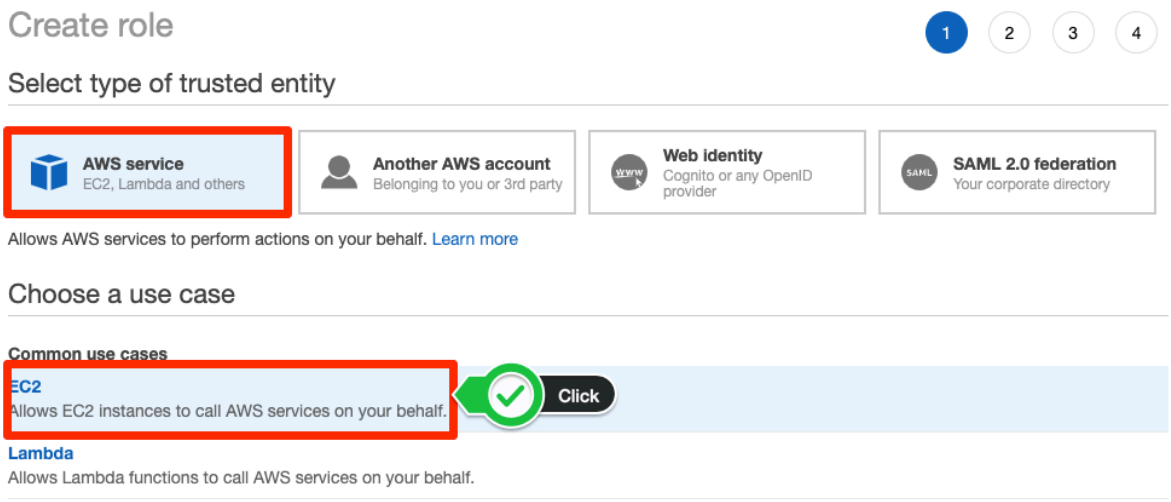


3. Upload any file (sample: aws-logo-picture) in the S3 bucket.



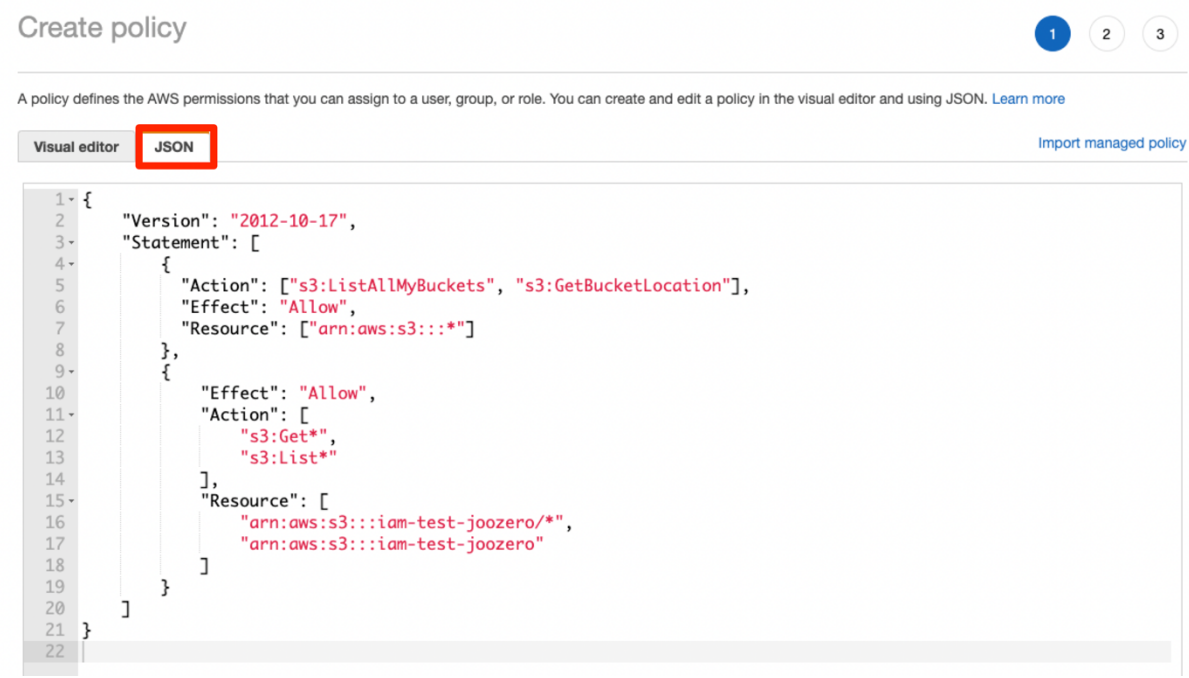
4. Create one more bucket named iam-test-other-user_name. Upload any file in the S3 bucket.

5. Move on to the IAM console to create IAM role for EC2 instance. Click **Roles** on the left side of the IAM console and click **Create role** button. On first step, choose **EC2** for trusted entity and click **Next: Permissions**. IAM Role can be applied to many AWS services including EC2 and Lambda, as well as other AWS accounts, Web identity, and SAML 2.0 federation.



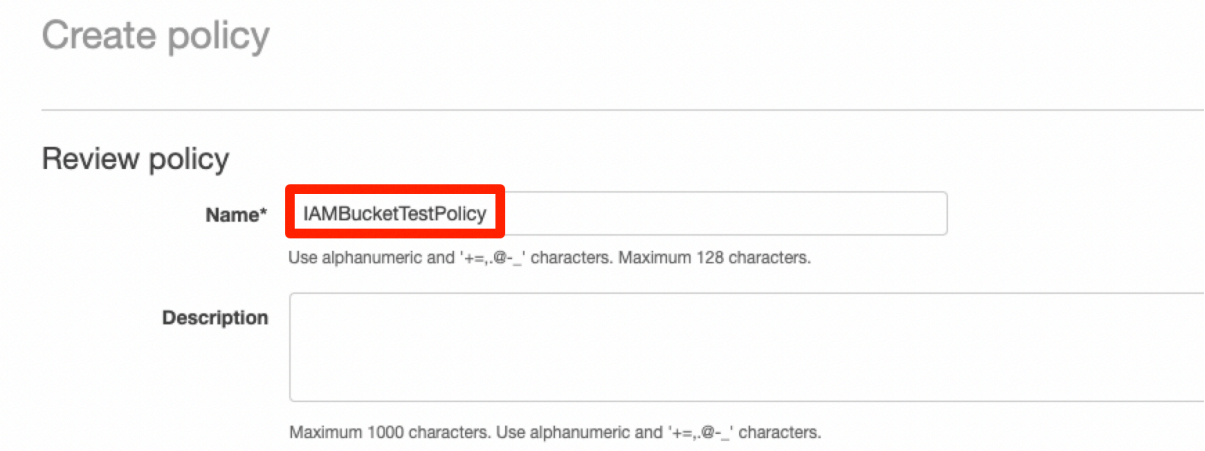
6. In step 2, click **Create policy** to create a policy to attach to the EC2 instance's role. On the **JSON** tab, paste the policy below and click **Next: Tags**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
        "Effect": "Allow",
        "Resource": ["arn:aws:s3:::*"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::iam-test-user_name/*",
                "arn:aws:s3:::iam-test-user_name"
            ]
        }
    ]
}
```

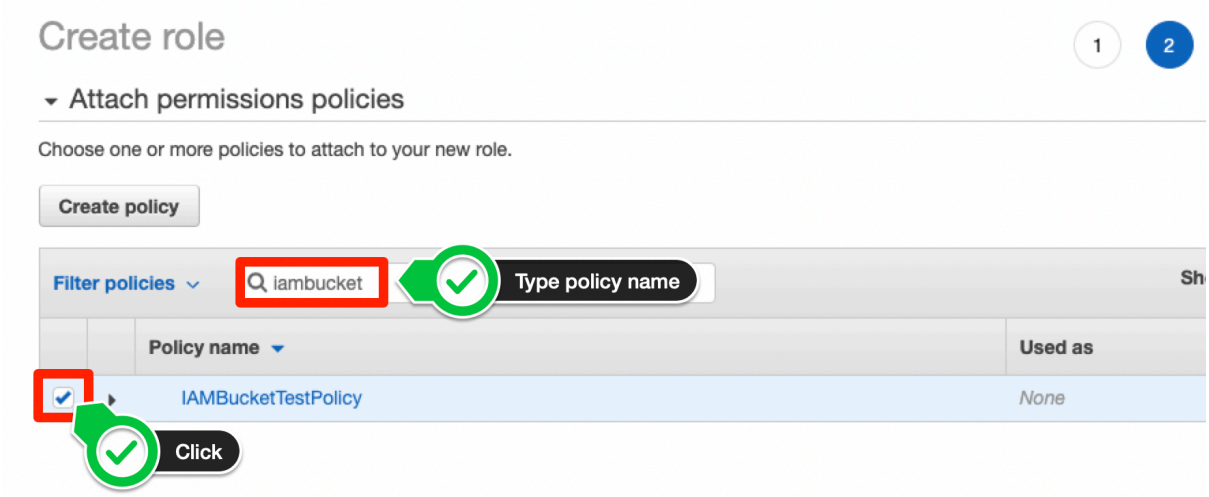


In the json file above, make sure to change the value in the **Resource** to **the name of the S3 bucket you created**.

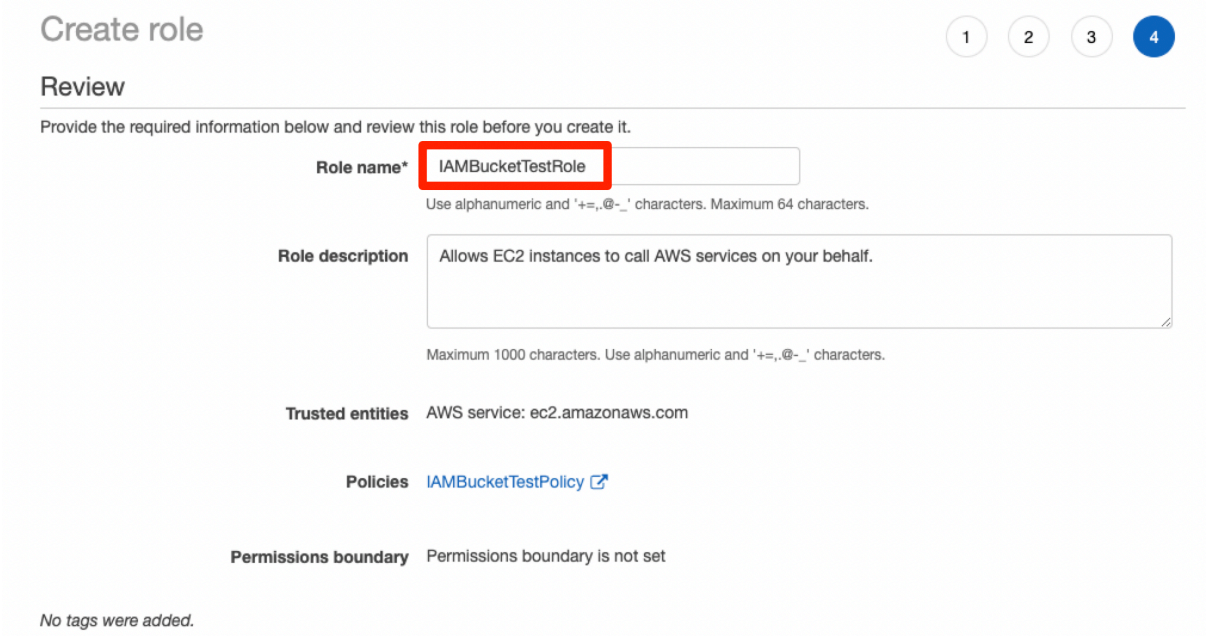
6. To skip adding tags operation, click **Next:Review** and type IAMBucketTestPolicy in the **Name** input box. Click **Create policy** to attach to IAM role.



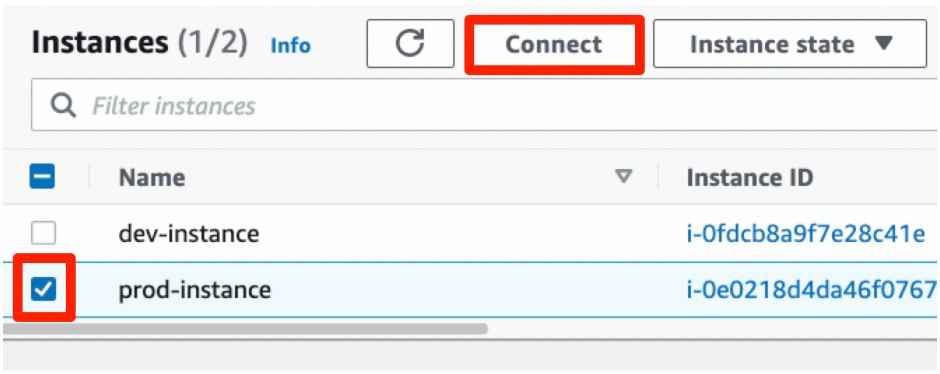
7. Back to the page where you were creating IAM Role, press the refresh button on the right corner and type IAMBucketTestPolicy. If you see the policy you just created in the list below, select it and go to the next step.



8. In step 4, type IAMBucketTestRole and create role.



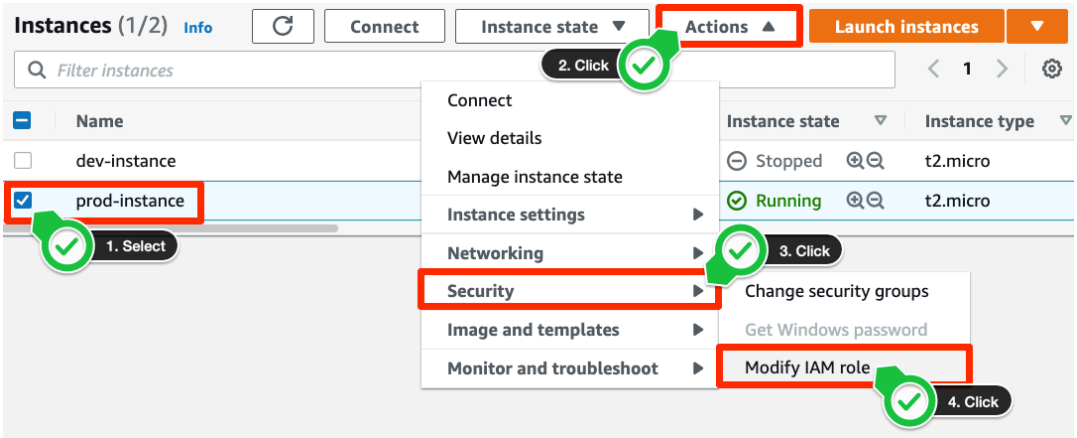
9. Move on to the **EC2 console** to attach IAM role on the EC2 instance. At this time, if you don't have EC2 instances created in the previous lab, check the top right to see whether AWS Region is properly set up. Select prod-instance, then click the **Connect** button.



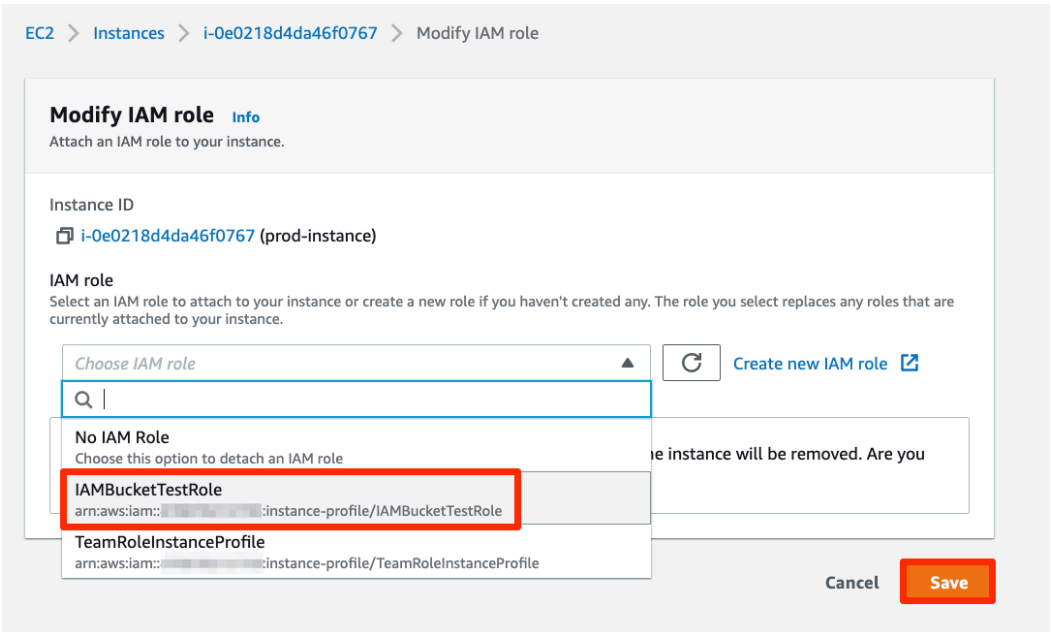
10. Connect to instance by using EC2 Instance Connect option, and click **Connect** button. Then the terminal will come out as below. Type aws s3 ls and you cannot get S3 bucket lists.

```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
2 package(s) needed for security, out of 6 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-16-4 ~]$ aws s3 ls

An error occurred (AccessDenied) when calling the ListBuckets operation: Access Denied
[ec2-user@ip-172-31-16-4 ~]$
```

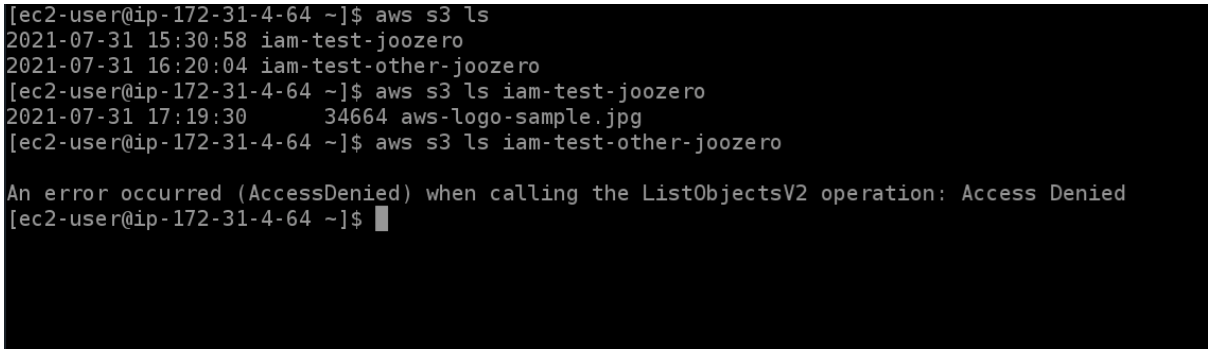
11. Back to the **Instances** page and select prod-instance. Click **Actions** button and select **Security** and click **Modify IAM role**.



12. In IAM role, select IAMBucketTestRole and click **Save** button to attach IAM role to EC2 instance.



13. Again, connect to the EC2 instance and type aws s3 ls again. Now you can see S3 bucket lists. Also, you can see object list placed in iam-test-user_name but not in iam-test-other-user_name because you don't have IAM policy for this bucket.




```
aws s3 ls

aws s3 ls iam-test-user_name

aws s3 ls iam-test-other-user_name
```

Great Job! You finished all hands on lab about IAM