AWS IAM

IAM

AWS IAM (Identity and Access Management) is a service provided by Amazon Web Services (AWS) that helps you manage access to your AWS resources. It's like a security system for your AWS account.

IAM allows you to create and manage users, groups, and roles. Users represent individual people or entities who need access to your AWS resources. Groups are collections of users with similar access requirements, making it easier to manage permissions. Roles are used to grant temporary access to external entities or services.

With IAM, you can control and define permissions through policies. Policies are written in JSON format and specify what actions are allowed or denied on specific AWS resources. These policies can be attached to IAM entities (users, groups, or roles) to grant or restrict access to AWS services and resources.

IAM follows the principle of least privilege, meaning users and entities are given only the necessary permissions required for their tasks, minimizing potential security risks. IAM also provides features like multi-factor authentication (MFA) for added security and an audit trail to track user activity and changes to permissions.

By using AWS IAM, you can effectively manage and secure access to your AWS resources, ensuring that only authorized individuals have appropriate permissions and actions are logged for accountability and compliance purposes.

Overall, IAM is an essential component of AWS security, providing granular control over access to your AWS account and resources, reducing the risk of unauthorized access and helping maintain a secure environment.

Components of IAM

Users: IAM users represent individual people or entities (such as applications or services) that interact with your AWS resources. Each user has a unique name and security credentials (password or access keys) used for authentication and access control.

Groups: IAM groups are collections of users with similar access requirements. Instead of managing permissions for each user individually, you can assign permissions to groups, making it easier to manage access control. Users can be added or removed from groups as needed.

Roles: IAM roles are used to grant temporary access to AWS resources. Roles are typically used by applications or services that need to access AWS resources on behalf of users or other services. Roles have associated policies that define the permissions and actions allowed for the role.

Policies: IAM policies are JSON documents that define permissions. Policies specify the actions that can be performed on AWS resources and the resources to which the actions apply. Policies can be attached to users, groups, or roles to control access. IAM provides both AWS managed policies (predefined policies maintained by AWS) and customer managed policies (policies created and managed by you).

Interview Questions

Q: What is AWS IAM, and why is it important?

A: AWS IAM (Identity and Access Management) is a service provided by Amazon Web Services that helps you control access to your AWS resources. It allows you to manage user identities, permissions, and policies. IAM is important because it enhances security by ensuring that only authorized individuals or entities have access to your AWS resources, helping you enforce the principle of least privilege and maintain a secure environment.

Q: What is the difference between IAM users and IAM roles?

A: IAM users represent individual people or entities that need access to your AWS resources. They have their own credentials and are typically associated with long-term access. On the other hand, IAM roles are used to grant temporary access to AWS resources, usually for applications or services. Roles have associated policies and can be assumed by trusted entities to access resources securely.

Q: What are IAM policies, and how do they work?

A: IAM policies are JSON documents that define permissions. They specify what actions are allowed or denied on AWS resources and can be attached to IAM users, groups, or roles. Policies control access by matching the actions requested by a user or entity with the actions allowed or

denied in the policy. If a requested action matches an allowed action in the policy, access is granted; otherwise, it is denied.

Q: What is the principle of least privilege, and why is it important in IAM?

A: The principle of least privilege states that users should be granted only the permissions necessary to perform their tasks and nothing more. It is important in IAM because it minimizes the risk of unauthorized access and limits the potential damage that could be caused by a compromised account. Following the principle of least privilege helps maintain a secure environment by ensuring that users have only the permissions they need to perform their job responsibilities.

Q: What are an AWS managed policy?

A: An AWS managed policy is a predefined policy created and managed by AWS. These policies cover common use cases and provide predefined permissions for specific AWS services or actions. AWS managed policies are maintained and updated by AWS, ensuring they stay up to date with new AWS services and features. They can be attached to IAM users, groups, or roles in your AWS account.

AWS IAM (Identity and Access Management)

What is IAM?

IAM is a web service that helps you securely control access to AWS resources. It allows you to manage:

Who can access your AWS account (users, roles)

What actions they can perform (read, write, delete)

Which resources they can access (S3 buckets, EC2, etc.)

FREE — IAM is a global AWS service and comes at no additional cost.

Key Features of IAM

Feature	Description
Granular permissions	Fine-tuned access using policies
Multi-factor auth (MFA)	Extra layer of login security
Temporary credentials	Use IAM roles for short-lived access
Centralized control	Manage all permissions from one place
Supports least privilege	Only give the minimum required permissions

IAM Users

Represents a person or application in your AWS account

Has credentials (username, password, and/or access keys)

You assign policies to define what they can access

Hands-On: Create IAM User

1. Go to IAM > Users > Add Users
2. Enter name: `dev-user`
3. Select Access type:

   AWS Management Console

   Programmatic Access (CLI/SDK)
4. Set password or generate access key
5. Attach permissions:

   Choose `Attach policies directly`

Add policy: `AmazonS3ReadOnlyAccess` (for testing)

6. Complete and save credentials

✅ Done! You now have a new IAM user.

IAM Groups

A collection of IAM users

Useful to apply same policy to multiple users

Hands-On: Create Group

1. Go to IAM > User groups > Create group
2. Name: `DevTeam`
3. Attach policy: `AmazonEC2ReadOnlyAccess`
4. Add existing users (like `dev-user`)

Now `dev-user` inherits the group's permissions.

IAM Roles

IAM Role is an identity with permissions, but without credentials

Used for temporary access

Trusted entities (like EC2, Lambda, another AWS account) assume the role

Use Cases:

EC2 accessing S3

Lambda accessing DynamoDB

Cross-account access

Hands-On: Create EC2 Role to access S3

1. Go to IAM > Roles > Create Role
2. Trusted entity: Choose `AWS service` > `EC2`
3. Attach policy: `AmazonS3ReadOnlyAccess`
4. Name the role: `EC2S3AccessRole`
5. Launch a new EC2 instance and under IAM role, select `EC2S3AccessRole`

Now EC2 can access S3 buckets using this role (no credentials needed).

IAM Policies

A JSON document that defines permissions

Attached to users, groups, or roles

Example Policy:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::my-bucket"
    }
  ]
}
```

This allows listing objects in `my-bucket`.

Hands-On: Create Custom Policy

1. Go to IAM > Policies > Create Policy
2. Choose JSON tab and paste the above
3. Click Next, name it: `ListMyBucketPolicy`
4. Attach it to a user, group, or role

Summary Chart

Component	Description	Use Case
IAM User	Individual with login credentials	Dev, Admin
IAM Group	Group of users sharing permissions	Teams
IAM Role	Temp access without credentials	EC2, Lambda
Policy	JSON rule set defining access	Attach to users, groups, roles

BONUS: IAM Best Practices

Enable MFA for root and users

Use roles instead of access keys for services

Follow principle of least privilege

Rotate access keys regularly

IAM Policy Structure

IAM policies are written in JSON format and consist of the following key elements:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow" | "Deny",
      "Action": "service:action",
      "Resource": "arn:aws:service:region:account-id:resource",
      "Condition": {
        "ConditionOperator": {
          "ConditionKey": "ConditionValue"
        }
      }
    }
  ]
}
```

Elements Explained

Element	Required	Purpose
Version	Yes	Specifies the policy language version. Always use "2012-10-17"
Statement	Yes	One or more permissions blocks
Effect	Yes	Allow or Deny access
Action	Yes	API operations allowed/denied (e.g., s3:PutObject)
Resource	Yes (in most cases)	The AWS resource(s) to which the action applies
Condition	Optional	Add logic like IP checks, time, MFA requirement

Example 1: Allow S3 Read-Only Access

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket",
        "arn:aws:s3:::my-bucket/"
      ]
    }
  ]
}
```

This allows read access to a specific S3 bucket and its contents.

Example 2: Deny access unless using MFA

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "",
      "Resource": "",
      "Condition": {
        "BoolIfExists": {
          "aws:MultiFactorAuthPresent": "false"
        }
      }
    }
  ]
}
```

Denies all actions unless the user is authenticated using MFA.

Example 3: Allow EC2 actions only from corporate IP

```json
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": "ec2:",
      "Resource": "",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "203.0.113.0/24"
        }
      }
    }
  ]
}
```

Grants EC2 access only if request comes from the corporate IP range.

Action Format (Service + Operation)

Action	Meaning
s3:PutObject	Upload object to S3
ec2:StartInstances	Start EC2 instances
dynamodb:Query	Query a DynamoDB table

You can use `` as a wildcard:

`s3:` = all S3 actions

`ec2:Describe` = all EC2 describe actions

Resource Format (ARN)

Most policies target specific AWS resources using Amazon Resource Names (ARNs).

General format:

```
arn:aws:<service>:<region>:<account-id>:<resource-type>/<resource-name>
```

Example:

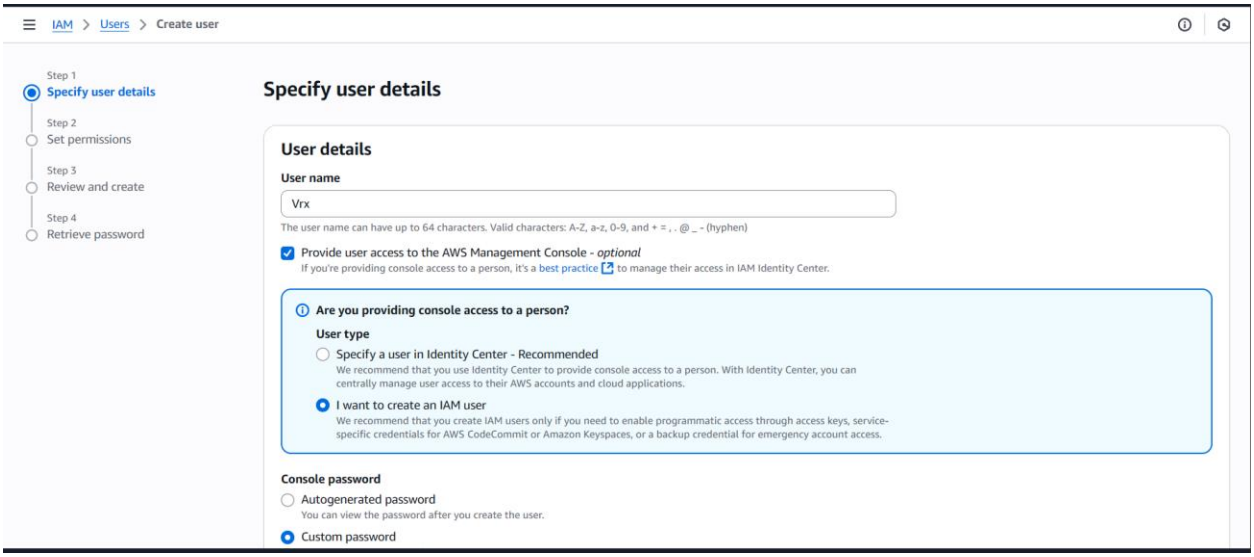
`arn:aws:s3:::my-bucket` (S3 bucket)

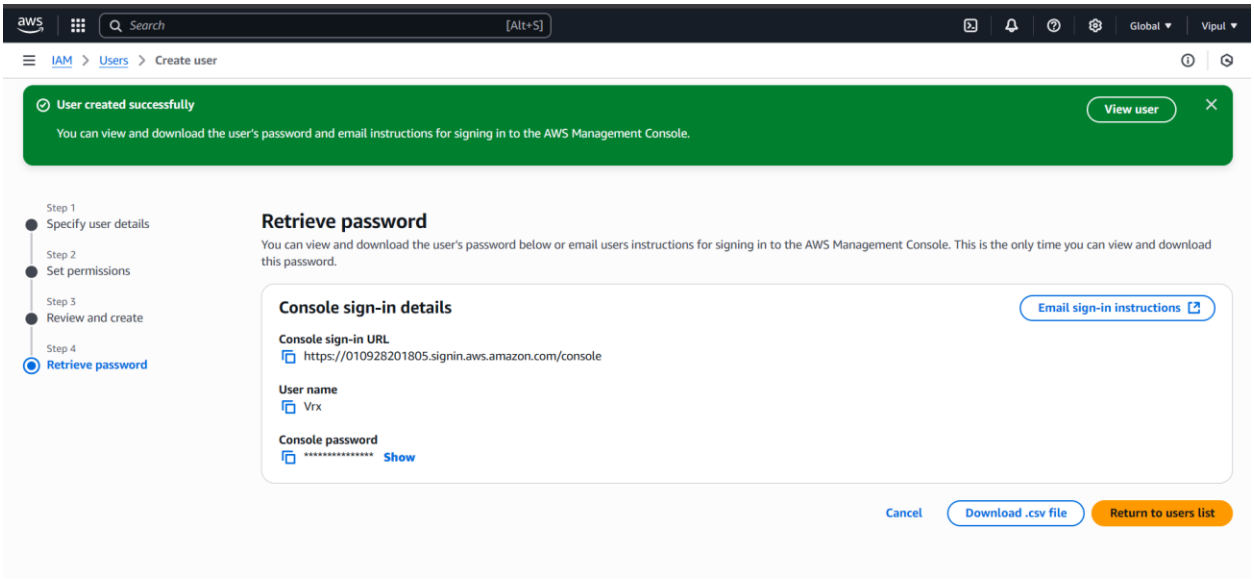
`arn:aws:ec2:us-east-1:123456789012:instance/i-1234567890abcdef0` (EC2 instance)

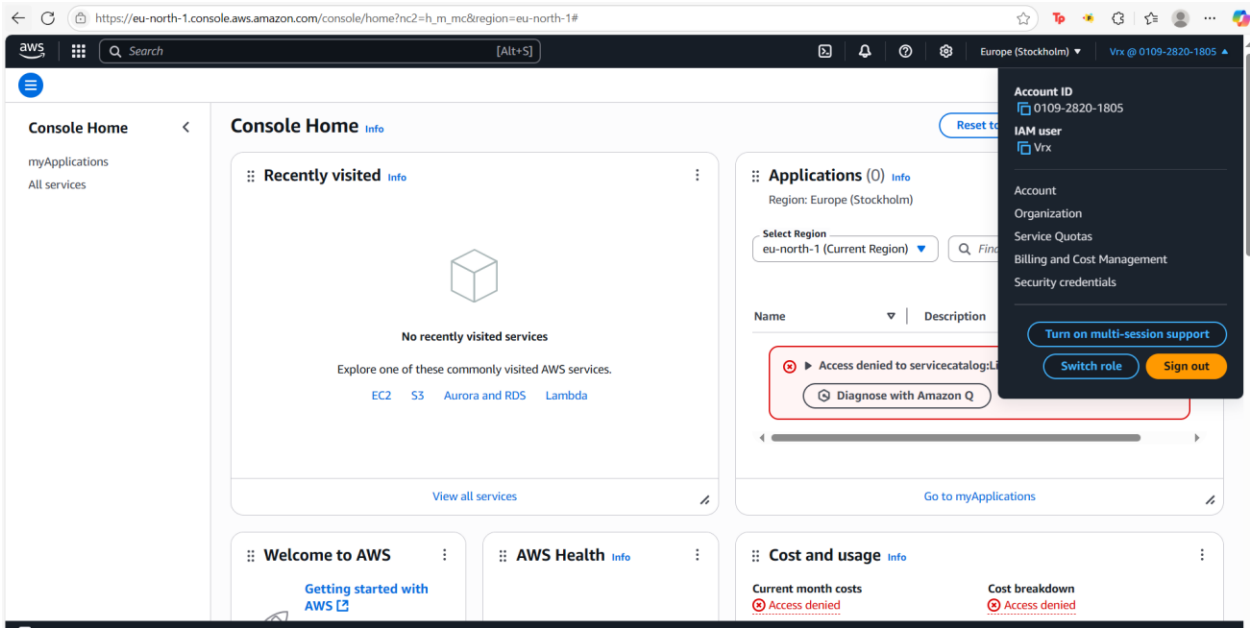
How to Create a Policy (Hands-On)

1. Go to IAM > Policies > Create Policy
2. Choose JSON tab
3. Paste one of the above examples
4. Click Next > Add tags (optional) > Review
5. Give it a name: e.g., `ReadOnlyS3Policy`
6. Click Create Policy
7. Attach it to a user, group, or role

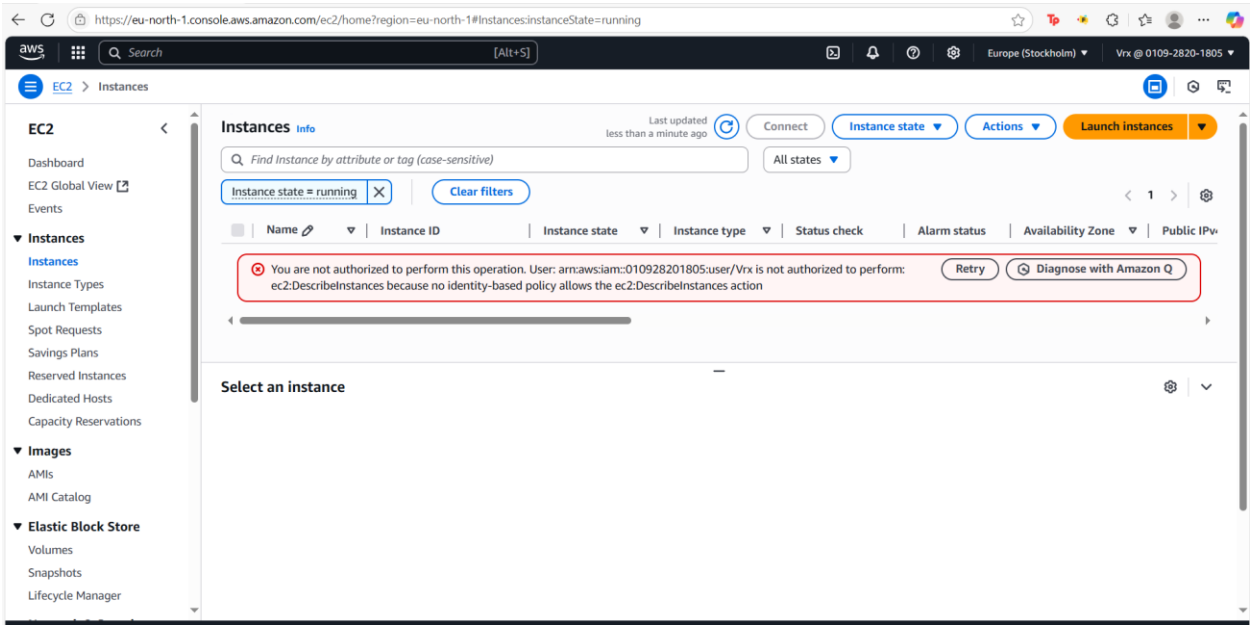
EXAMPLE: Created a IAM user and given policy in order to access the resource.





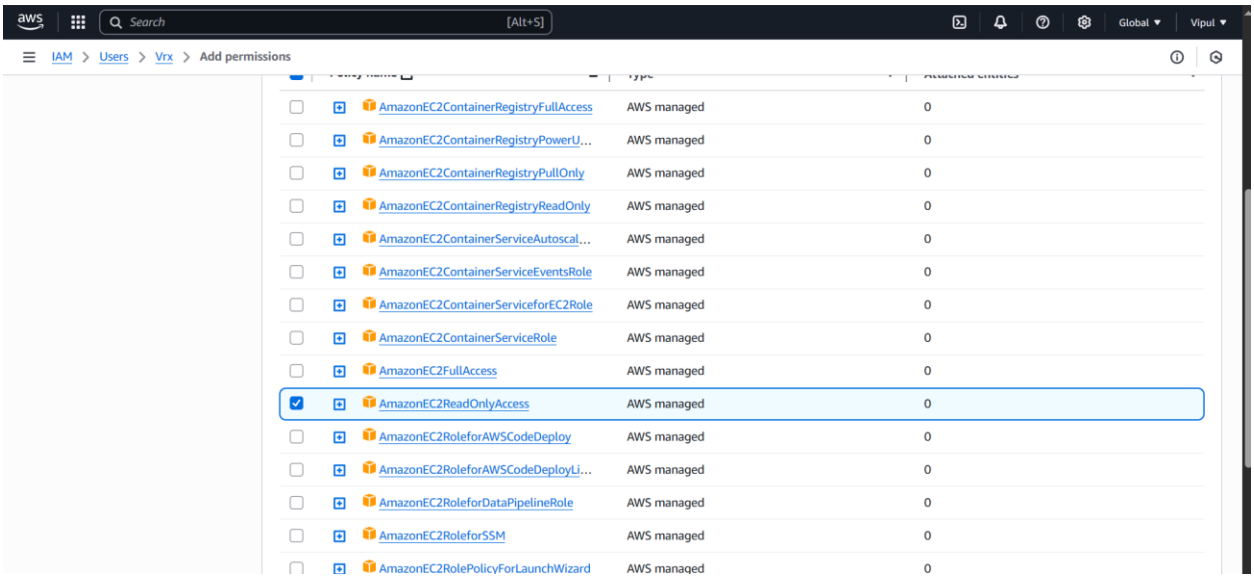


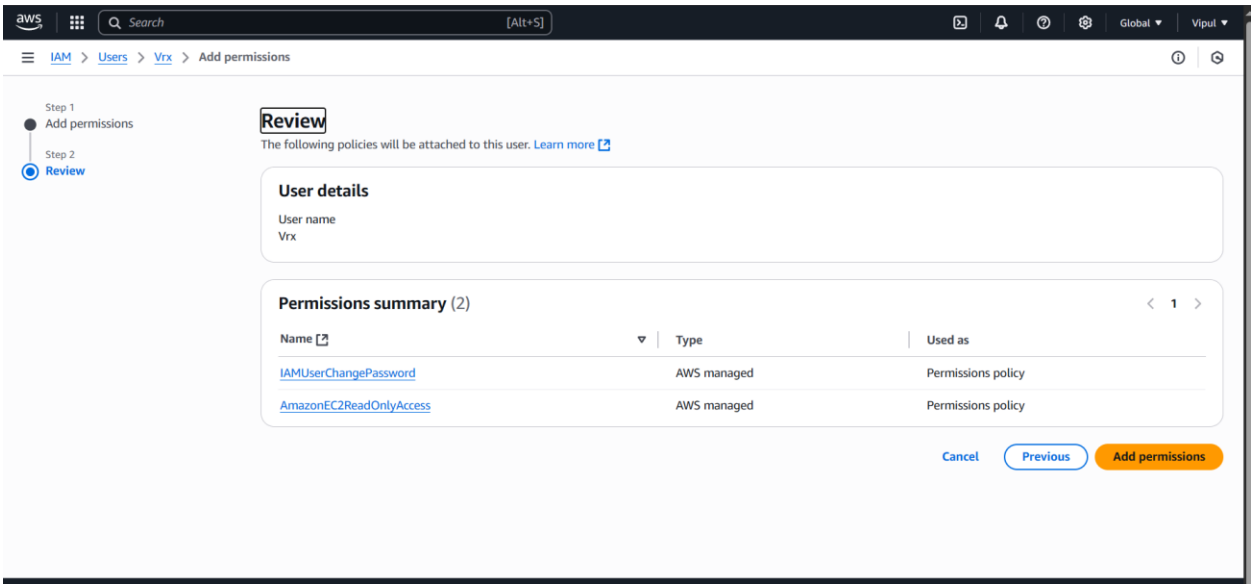
account id is present in the Url provided in the .csv file

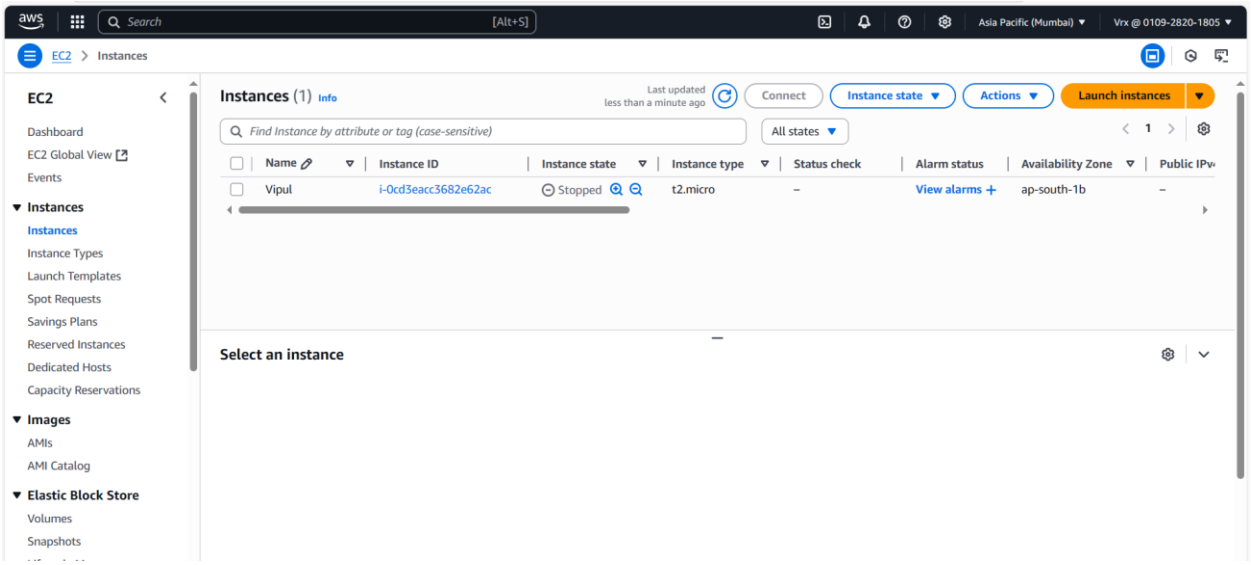


Currently no policy has been provided to the user so I cannot access any resources.

Now I am authorizing user to access the resources.







the resource is accessible now
NOTE: The user and resource need to be in same region in order to access it.

https://policysim.aws.amazon.com/home/index.jsp#
this is useful link will get you later

Users are used for authentication and Policies are used for authorization.