

@id -> marks id as private key
@generatedValue(strategy=GenerationType.Identity) -> The database will autogenerate the id

Configuration management tools are idempotent

Server templating tool -> Docker -> create custom image of a virtual machine or a container.
These images contain all the required software and dependencies installed once, eliminates the need for installing software -> custom AMIs in AWS, Docker images in dockerhub -> immutable infrastructure unlike configuration management tools.

Terraform -> PROVISIONING infrastructure -> build manage and destroy configuration -> deploy infrastructures across multiple platforms like pvt or pub cloud (AWS, GCP, Azure)

How does Terraform manage infra across so many platforms? -> Providers -> make manage 3rd party platforms through their API

Terraform uses HCL (hashicorp config language)
Can be maintained in a version control
Code is declarative -> terraform will take care of going from current state to desired state without worries how to get there.

Init -> identifies providers
Plan -> plans to get to desired state

Resource -> every object terraform manages

Block:



```
local.tf

resource "local_file" "pet" {
  filename = "/root/pets.txt"
  content = "We love pets!"
}
```

Block -> defined within curly braces -> configuration data in key value format
/root -> new directory -> we will create HCL configuration files - local.tf -> inside this define resource block.

Here, local_file is the resource type. Resource type provides 2 bits info -> local - provider, file - type of resource.

"pet" is the resource name.

Within braces, we write arguments for resource.

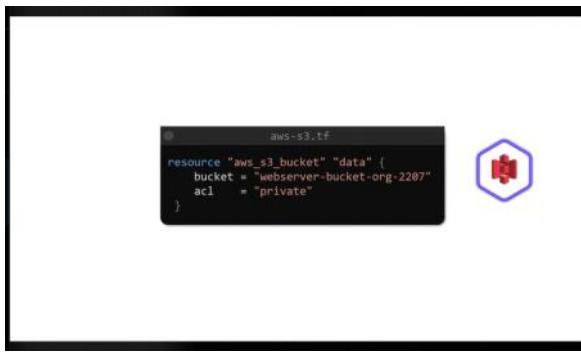
1st argument - filename -> absolute path.
2nd argument - content -> value.

Now, we have complete HCL configuration file by the name pet.txt
Terraform init will install a plugin called local - so we are ready to create the resource.



```
aws-ec2.tf

resource "aws_instance" "webserver" {
  ami      = "ami-0c2f25c1f66a1ffad"
  instance_type = "t2.micro"
}
```



📘 What Are File Permissions?

In Linux/Unix systems, **file permissions** control **who can read, write, or execute** a file or directory.

There are **three types of permissions**:

- **r** = read (view the contents)
- **w** = write (modify the contents)
- **x** = execute (run the file as a program)

And **three types of users**:

1. **Owner** – the user who owns the file
2. **Group** – users in the same group as the file
3. **Others** – everyone else

💻 Numeric (Octal) Permission Codes

Each permission is represented by a number:

Permission	Binary	Decimal
r	100	4
w	010	2
x	001	1
-	000	0

So, you **add** the values to get the permission for each user type.

Example: chmod 754 file.txt

Breakdown:

- **7** = 4 (read) + 2 (write) + 1 (execute) → rwx (owner)
- **5** = 4 (read) + 0 (no write) + 1 (execute) → r-x (group)
- **4** = 4 (read) + 0 + 0 → r-- (others)

So the permission string is: rwxr-xr--

💻 Symbolic Representation

You might also see permissions like this when you run ls -l:

-rwxr-xr-- 1 user group file.txt

- The first character (-) indicates it's a file (could be d for directory).
- The next 9 characters are grouped as:
 - rwx → owner
 - r-x → group
 - r-- → others

✍ Common Permission Codes

Code	Meaning	Symbolic
777	Full access to all	rwxrwxrwx
755	Owner: all, others: read + execute	rwxr-xr-x
644	Owner: read + write, others: read	rw-r--r--
700	Owner only access	rwx-----

Ah, I see where the confusion is coming from — you're referring to **four-digit file permission codes** like "0700" or "0755". Let's clarify what the **first digit** means in a **four-digit permission code**.

13 Four-Digit File Permission Code: 0###

- The **last three digits** (###) are the standard permissions for:
 - Owner**
 - Group**
 - Others**
- The **first digit** (the leading 0 in 0700) is **not a user** — it represents **special permission bits**.

1 Special Permission Bits (First Digit)

Value	Meaning
0	No special permissions
1	Sticky bit
2	Setgid (Set group ID)
4	Setuid (Set user ID)

These can be **combined** by adding the values:

- $7 = 4 + 2 + 1 \rightarrow \text{Setuid} + \text{Setgid} + \text{Sticky bit}$

1 Example: 4755

- 4 → Setuid is set
- 7 → Owner: rwx
- 5 → Group: r-x
- 5 → Others: r-x

This means:

- The file runs with the **owner's privileges** (Setuid).
- Normal permissions apply to group and others.

1 Common Use Case

- /usr/bin/passwd has permission -rwsr-xr-x (or 4755)
 - The s in rws means **Setuid** is active.
 - This allows users to change their password (which modifies /etc/shadow) even though they don't have write access to it — because the binary runs with root privileges.

So to answer your question directly:

Who is the "other user" in a 4-digit permission like 0700?

There is **no fourth user**. The **first digit** is for **special permissions**, not a user category.

Would you like a visual breakdown or a cheat sheet for these special bits?

From <<https://www.microsoft365.com/chat?version=19.2511.33071.0&capabilities=interopPromise.suspendOnClose.autoStart&client-request-id=c1e9d5db-187c-6b44-bfac-7f41f19acd51>>

Now, local.tf -> It is a configuration file.
Config. dir. Can contain more than one cfg file.

Terraform's plugin-based architecture supports hundreds of infrastructure platforms, and the providers are distributed by HashiCorp via the [Terraform Registry](#).

Parameterizing with Variable instead of hardcoding values

Variables.tf

```
variable "filename" {
  default = "/root/pets.txt"
}
```

```
variable "filename" {
  default = "/root/pets.txt"
}

variable "content" {
  default = "We love pets!"
}

variable "prefix" {
  default = "Mrs"
}

variable "separator" {
  default = "."
}

variable "length" {
  default = "1"
}
```

Main.tf:

```
# main.tf
resource "local_file" "pet" {
  filename = var.filename
  content  = var.content
}

resource "random_pet" "my-pet" {
  prefix    = var.prefix
  separator = var.separator
  length    = var.length
}

# variables.tf
variable "filename" {
  default = "/root/pets.txt"
}

variable "content" {
  default = "We love pets!"
}
```

Creating an EC2 Instance with Variables

Here's an additional example of using input variables to create a configuration. This configuration demonstrates the same variable usage pattern, even if you're not familiar with AWS:

```
resource "aws_instance" "webserver" {
  ami          = var.ami
  instance_type = var.instance_type
}

variable "ami" {
  default = "ami-0edab43b6fa892279"
}

variable "instance_type" {
  default = "t2.micro"
}
```

Variable Blocks

Apart from default, variable block can also contain several parameters.

```
variable "filename" {
  default      = "/root/pets.txt"
  type         = string
  description = "The path of the local file"
}

variable "content" {
  default      = "I love pets!"
  type         = string
  description = "The content of the file"
}

variable "prefix" {
  default      = "Mrs"
  type         = string
  description = "The prefix to be set"
}

variable "separator" {
  default = ","
}
```

List Variables

A list is an ordered collection of values where each value is a variable. Consider a variable that uses a list of prefixes:

```
variable "prefix" {
  default = ["Mr", "Mrs", "Sir"]
  type    = list(string)
}

resource "random_pet" "my-pet" {
  prefix = var.prefix[0]
}
```

In this configuration:

- `var.prefix[0]` returns "Mr"
- `var.prefix[1]` returns "Mrs"
- `var.prefix[2]` returns "Sir"

Map Variables

Maps allow you to define key-value pairs for storing related data. For example:

```
variable "file_content" {
  type    = map(string)
  default = {
    "statement1" = "We love pets!"
    "statement2" = "We love animals!"
  }
}
```

To reference a specific value from this map in a resource, use the key:

```
resource "local_file" "my-pet" {
  filename = "/root/pets.txt"
  content  = var.file_content["statement2"]
}
```

List and Map Type Constraints

You can enforce type constraints on lists and maps to ensure they contain the correct types of values.

```
variable "prefix" {
  default = ["Mr", "Mrs", "Sir"]
  type    = list(string)
}

variable "numbers" {
  default = [1, 2, 3]
  type    = list(number)
}
```

```
variable "cats" {
  default = {
    "color" = "brown"
    "name" = "bella"
  }
  type = map(string)
}

variable "pet_count" {
  default = {
    "dogs" = 3
    "cats" = 1
    "goldfish" = 2
  }
  type = map(number)
}
```

Using variables in terraform

If a variable does not have a default value or if you want to override an existing default, Terraform will prompt you for a value during `terraform apply`.

Providing Variable Values Interactively and via the Command Line

If a variable does not have a default value or if you want to override an existing default, Terraform will prompt you for a value during `terraform apply`. To streamline automation and avoid interactive prompts, you can pass values using the `-var` flag. You can supply multiple `-var` flags as needed:

```
$ terraform apply -var "filename=/root/newfile.txt" -var "content=Hello, Terraform!"
```

Alternatively, you can set environment variables by prefixing the variable name with `TF_VAR_`. For example, you can configure your shell as follows:

```
$ export TF_VAR_filename="/root/pets.txt"
$ export TF_VAR_content="We love pets!"
$ export TF_VAR_prefix="Mrs"
$ export TF_VAR_separator=". "
$ export TF_VAR_length="2"
$ terraform apply
```

Using Variable definition files

Why Use `terraform.tfvars`?

You don't have to use `terraform.tfvars` — but it's very useful when:

1. You want to override default values

Instead of editing `variables.tf`, you can just assign values in `terraform.tfvars`.

2. You want to separate logic from data

Keep your infrastructure logic (`main.tf`) clean and put actual values in a separate file.

3. You want to switch environments easily

You can create:

- `dev.tfvars`
- `prod.tfvars`
- `test.tfvars`

Then run:

```
terraform apply -var-file="prod.tfvars"
```

This makes your setup **modular and scalable**.

Example Without `terraform.tfvars`

You can do everything in `variables.tf` like this:

```
Terraform
variable "filename" {
  default = "games.txt"
}

variable "content" {
  default = "FIFA 21"
}
```

Show more lines

And use `var.filename` and `var.content` in `main.tf`.

This works perfectly for **simple or static setups**.

So When Should You Use `terraform.tfvars`?

Use it when:

- You want to **change values without editing code**
- You have **multiple environments**
- You want to **keep secrets or sensitive values** out of version-controlled files (you can even use `.tfvars` files in `.gitignore`)

When managing many variables, it becomes practical to store their values in a dedicated variable definition file. These files typically have a `.tfvars` or `.tfvars.json` extension.

Using Variable Definition Files

When managing many variables, it becomes practical to store their values in a dedicated variable definition file. These files typically have a `tfvars` or `tfvars.json` extension. For example, you can create a file named `terraform.tfvars` with the following contents:

```
filename = "/root/pets.txt"
content = "We love pets!"
prefix = "Mrs"
separator = "."
length = "2"
```

Terraform automatically loads files named `terraform.tfvars`, `terraform.tfvars.json`, or files with extensions like `auto.tfvars` or `auto.tfvars.json`. If you use a differently named file (e.g., `variables.tfvars`), be sure to specify it explicitly with the `--var-file` flag:

```
$ terraform apply --var-file="variables.tfvars"
```

This approach centralizes your variable definitions and simplifies the management of Terraform environments.

Resource Attributes

Consider a scenario where you want to use the output of one resource as an input for another.

1. Create a file resource

```
1 resource "local_file" "games" {
2   filename = "${path.module}/favorite-games.txt"
3   content = "FIFA 21"
4 }
```

2. Use its attribute in another resource

```
1 resource "null_resource" "print_file_path" {
2   provisioner "local-exec" {
3     command = "echo File created at ${local_file.games.filename}"
4   }
5 }
```

Here:

- `local_file.games.filename` is an **attribute** of the `local_file` resource.
- It's used as input in the `null_resource` to print the file path.

when you pass the output of one resource (like a random pet) to another resource (such as a local file), Terraform understands that the random pet must be created before the local file -> implicit dependency

Below is an example of an implicit dependency:

```
resource "local_file" "pet" {
  filename = var.filename
  content  = "My favorite pet is ${random_pet.my-pet.id}"
}

resource "random_pet" "my-pet" {
  prefix   = var.prefix
  separator = var.separator
  length    = var.length
}
```

Output Variables

Terraform output variables are a powerful feature that allow you to store the results of expressions from your configuration files for later use.

Below is an example configuration:

```
resource "local_file" "pet" {
  filename = var.filename
  content  = "My favorite pet is ${random_pet.my-pet.id}"
}

resource "random_pet" "my-pet" {
  prefix   = var.prefix
  separator = var.separator
  length    = var.length
}

output "pet-name" {
  value = random_pet.my-pet.id
  desc  = ""
}
```

Terraform State

Terraform maintains a state file, which is how it tracks that the resource is already provisioned.

After the initial successful `terraform apply`, an additional file named `terraform.tfstate` is created in the project directory.

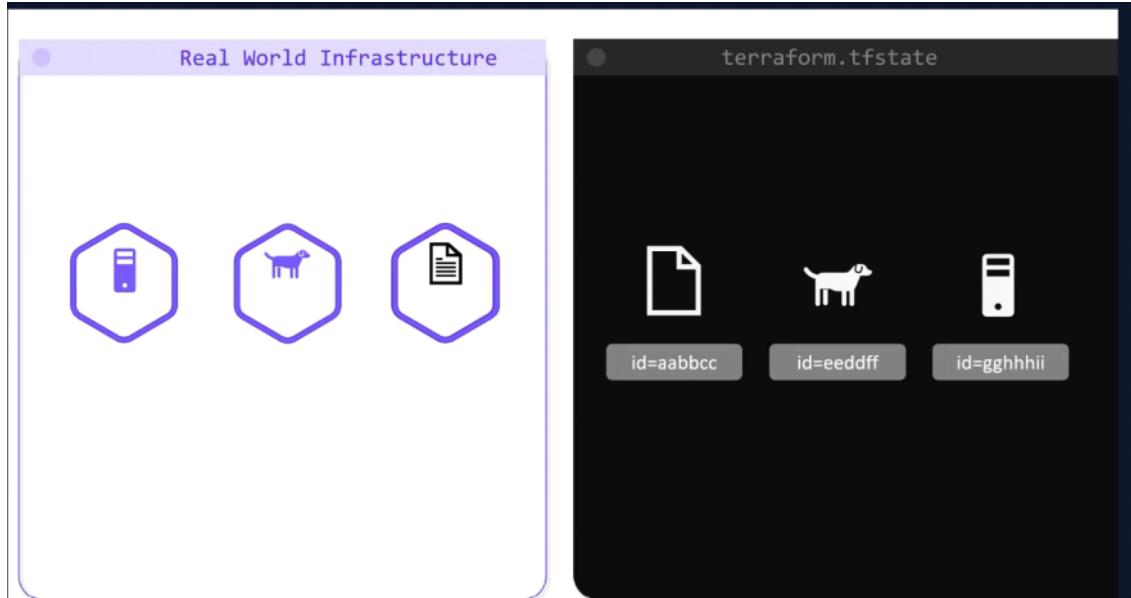
Inspecting `terraform.tfstate` reveals a detailed record of the infrastructure, including resource IDs, provider information, and resource attributes:

```
{  
  "version": 4,  
  "terraform_version": "0.13.0",  
  "serial": 1,  
  "lineage": "e35dde72-a943-de50-3c8b-1df8986e5a31",  
  "outputs": {},  
  "resources": [  
    {  
      "mode": "managed",  
      "type": "local_file",  
      "name": "pet",  
      "provider": "provider[\"registry.terraform.io/hashicorp/local\"]",  
      "instances": [  
        {  
          "schema_version": 0,  
          "attributes": {  
            "content": "I love pets!",  
            "content_base64": null,  
            "directory_permission": "0777",  
            "file_permission": "0777",  
            "filename": "/root/pets.txt",  
            "id": "7eddb4fbfdcb108dd04692062bae39bd1e1b68",  
            "sensitive_content": null  
          },  
          "private": "bnVzBA=="  
        }  
      ]  
    }  
  ]  
}
```

Now, if we update configurations in `variables.tf`, running `terraform apply` causes Terraform to refresh the state and detect a difference between the new configuration and the existing state. Consequently, Terraform decides the resource must be replaced.

After applying these changes, Terraform deletes the old resource and creates a new one with a different unique ID. The updated `terraform.tfstate` now reflects the new state.

Now that the configuration file and the state file are in sync, any subsequent runs of `terraform apply` will report that no changes are necessary.



Challenges with Mutable Infrastructure

When updating software on a system:

- Dependencies must be met for a successful upgrade.
- If one or more servers (e.g., web server 3) have unmet dependencies—such as network issues, storage constraints, or compatibility differences—the upgrade might fail, leaving that server on an older version (e.g., version 1.18).
- After multiple update cycles, configuration drift can occur, where servers run different versions of software, increasing the complexity of troubleshooting and further updates.

Embracing Immutable Infrastructure

Immutable infrastructure takes a different approach. Instead of modifying existing servers, you provision new servers with the updated version (for example, replacing Nginx 1.17 with 1.18). Once the new servers are confirmed to run correctly, the old servers are decommissioned.

Benefits of Immutable Infrastructure

- Easier to version infrastructure and roll back to previous releases.
- Consistent environments reduce the complexity of governance and maintenance.
- Reliable deployment pipelines, as each release is a fresh creation rather than an update

Terraform and Immutable Infrastructure

Terraform exemplifies immutable infrastructure by default. For instance, if you update a resource block—changing the permission from "0777" to "0700"—Terraform destroys the original resource and creates a new one with the updated settings.

Revisiting our earlier example:

```
resource "local_file" "pet" {
  filename      = "/root/pets.txt"
  content       = "We love pets!"
  file_permission = "0700"
}
```

By default, Terraform replaces the resource.

By default, when Terraform updates a resource, it treats it as immutable. This means the existing resource is deleted before a new one is created with the updated configuration. For example, if you update the file permissions on a local file resource from 0777 to 0700 and then run `terraform apply`, Terraform will first delete the old file and then create a new one.

The `create_before_destroy` Rule

The `create_before_destroy` lifecycle rule instructs Terraform to create a new resource before deleting the old one. This is particularly useful when maintaining service availability is critical.

```
resource "local_file" "pet" {
  filename      = "/root/pets.txt"
  content       = "We love pets!"
  file_permission = "0700"

  lifecycle {
    create_before_destroy = true
  }
}
```

The `prevent_destroy` Rule

In some cases, you might want to ensure a resource is never accidentally deleted—even if a configuration change would normally force a replacement. Terraform allows you to achieve this using the `prevent_destroy` rule.

- The `ignore_changes` rule allows you to specify attributes that Terraform should ignore during state comparisons, accommodating external changes.

Datasource

```
$ cat /root/dog.txt
Dogs are awesome!

resource "local_file" "pet" {
  filename = "/root/pets.txt"
  content  = data.local_file.dog.content
}

data "local_file" "dog" {
  filename = "/root/dog.txt"
}
```



Resource	Data Source
Keyword: <code>resource</code>	Keyword: <code>data</code>
Creates, Updates, Destroys Infrastructure	Only Reads Infrastructure
Also called Managed Resources	Also called Data Resources

Traditional Loop Approach

In traditional scripting, such as with bash, you might use a loop to create multiple files. The example below creates three empty files (`pet1`, `pet2`, and `pet3`) in the `/root` directory:

```
#!/bin/bash
for i in {1..3}
do
    touch /root/pet${i}
done
```

After running the above script, listing the directory contents may produce:

```
$ ls -ltr /root/
-rw-r--r-- 1 root root 0 Sep  9 02:04 pet2
-rw-r--r-- 1 root root 0 Sep  9 02:04 pet1
-rw-r--r-- 1 root root 0 Sep  9 02:04 pet3
```

Creating Multiple Instances with a Static Count

One of the simplest ways to create multiple instances of a resource is by using a static count. Below is an example that creates three instances of a local file resource:

```
resource "local_file" "pet" {
  filename = var.filename
  count    = 3
}
```

The variable is defined as follows:

```
variable "filename" {
  default = "/root/pets.txt"
}
```

When you run `terraform plan`, Terraform plans to create three resources:

Here,
Terraform creates the same file three times rather than three unique files.

Each resource is indexed as `pet[0]`, `pet[1]`, and `pet[2]`. Although three resources are created, all instances share the same file name.

Creating Unique Resources by Using a List Variable

To generate unique resources, update the variable definition to a list and reference each element using `count.index`.

Here is the modified configuration:

```
resource "local_file" "pet" {
  filename = var.filename[count.index]
  count    = 3
}
```

Define the list variable as follows:

```
variable "filename" {
  default = [
    "/root/pets.txt",
    "/root/dogs.txt",
    "/root/cats.txt"
  ]
}
```

In this configuration:

- The first iteration (index 0) creates `/root/pets.txt`.
- The second iteration (index 1) creates `/root/dogs.txt`.
- The third iteration (index 2) creates `/root/cats.txt`.

After executing `terraform apply`, listing the `/root` directory produces:

```
$ ls /root
pets.txt
dogs.txt
cats.txt
```

This confirms that each resource is now unique.

A drawback of the previous approach is its fixed count.

Using count with a List

Traditionally, resources are created using the count meta-argument. For example:

```
resource "local_file" "pet" {
  filename = var.filename[count.index]
  count    = length(var.filename)
}
```

```
variable "filename" {
  default = [
    "/root/pets.txt",
    "/root/dogs.txt",
    "/root/cats.txt"
  ]
}
```

This configuration creates resources as a list. However, updating resources based on index positions can lead to unexpected behavior when the list order changes.

Transitioning to for_each

The `for_each` argument only supports a map or a set of strings – not a list of strings

Correcting the Configuration

There are two approaches to resolve this issue:

1. Change the variable type from a list to a set (sets do not allow duplicate elements).
2. Convert the list into a set in the resource block using Terraform's built-in `toset` function.

Below is an updated configuration that uses the `toset` function:

```
resource "local_file" "pet" {
  filename = each.value
  for_each = toset(var.filename)
}

variable "filename" {
  type    = list(string)
  default = [
    "/root/pets.txt",
    "/root/dogs.txt",
    "/root/cats.txt"
  ]
}
```

When you run `terraform plan` now, Terraform will indicate that three resources will be created

Updating Resources by Removing an Element

Let's simulate updating the configuration by removing an element. For example, removing `/root/pets.txt` from the list changes the configuration to:

```
resource "local_file" "pet" {
  filename = each.value
  for_each = toset(var.filename)
}

variable "filename" {
  type    = list(string)
  default = [
    "/root/dogs.txt",
    "/root/cats.txt"
  ]
}
```

Running `terraform plan` with this updated variable shows that only the resource associated with `/root/pets.txt` will be destroyed:

Specifying a Specific Provider Version

To enforce a particular version of the local provider (for example, version 1.4.0), include a `terraform` block with a `required_providers` sub-block. This block explicitly instructs Terraform which version to install. Consider the following configuration:

```
terraform {
  required_providers {
    local = {
      source  = "hashicorp/local"
      version = "1.4.0"
    }
  }
}

resource "local_file" "pet" {
  filename = "/root/pet.txt"
  content  = "We love pets!"
}
```

For developers like Max, Abdul, and Lee who require access to both AWS EC2 and S3 services, a common practice is to create an IAM group (e.g., "Developer Group") and attach policies such as AmazonEC2FullAccess and AmazonS3FullAccess.

From <<https://notes.kodekloud.com/docs/Terraform-Basics-Training-Course/Terraform-with-AWS/introduction-to-IAM>>

AWS services, such as an EC2 instance, do not inherently possess permissions to interact with other AWS resources (for example, accessing an S3 bucket).

From <<https://notes.kodekloud.com/docs/Terraform-Basics-Training-Course/Terraform-with-AWS/introduction-to-IAM>>

To enable an AWS service to interact with another resource, you create an IAM role.

From <<https://notes.kodekloud.com/docs/Terraform-Basics-Training-Course/Terraform-with-AWS/introduction-to-IAM>>

Creating Custom IAM Policies

In addition to AWS managed policies, you can create custom IAM policies to match specific operational requirements. For instance, if you want a user to be able to create and delete tags on an EC2 instance, you could define a custom policy as shown below:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags",
        "ec2:DeleteTags"
      ],
      "Resource": "*"
    }
  ]
}
```

This custom policy grants the user permission exclusively to perform the "ec2:CreateTags" and "ec2:DeleteTags" actions on any EC2 resources. Once created, this policy can be assigned within your AWS account to enforce specific permissions.

From <<https://notes.kodekloud.com/docs/Terraform-Basics-Training-Course/Terraform-with-AWS/introduction-to-IAM>>

IAM is a global service—unlike other AWS services, once you create an IAM object (such as a user or group), it is available in every region across your account.

From <<https://notes.kodekloud.com/docs/Terraform-Basics-Training-Course/Terraform-with-AWS/Demo-IAM>>

Attaching Policies to a User

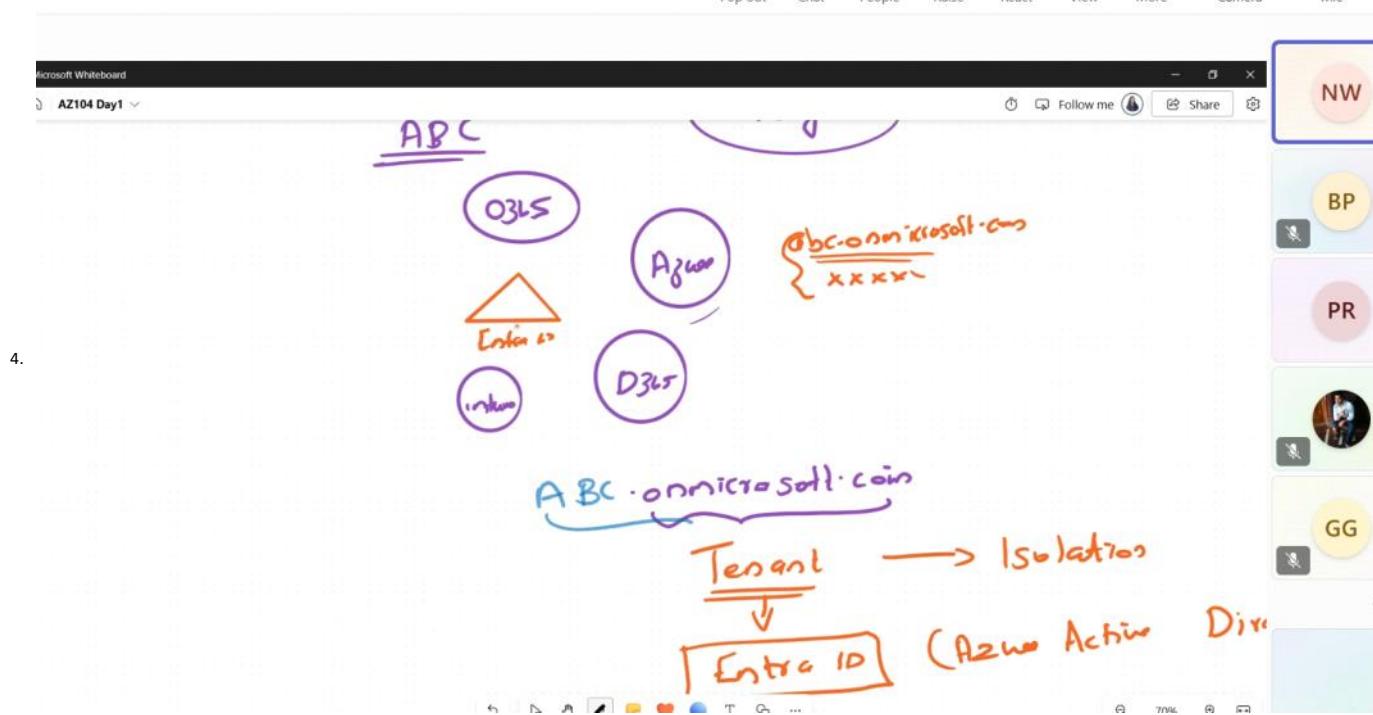
From <<https://notes.kodekloud.com/docs/Terraform-Basics-Training-Course/Terraform-with-AWS/Demo-IAM>>

Now, attach an additional policy that grants Lucy full administrative privileges. As the team lead, she needs unrestricted access to all account services. We will attach the AdministratorAccess Policy

From <<https://notes.kodekloud.com/docs/Terraform-Basics-Training-Course/Terraform-with-AWS/Demo-IAM>>

AZ 104 Day 1

1. Tenant -> renting services from ms. Each tenant is isolation
ABC.onmicrosoft.com -> entra id (Azure active directory) for authentication.
Enta id -> authentication and authorization
2. Every tenant has 1 entra id.
3. No group plicies in entra id no parent child rel



5. IAAS

Authenticatrion protocol ->SAML -> security assertion markup language

Wsfed

Open id connect

For authorization ->Oauth

On premise Ad

Turn on your camera to select this layout.

Describe Microsoft Entra ID Concepts

Concept	Description
<u>Identity</u>	An object that can be authenticated
<u>Account</u>	An identity that has data associated with it
<u>Microsoft Entra ID account</u>	An identity created through Microsoft Entra ID or another Microsoft cloud service
<u>Tenant/directory</u>	A dedicated and trusted instance. A tenant is automatically created when your organization signs up for a Microsoft cloud service subscription. <ul style="list-style-type: none">• Additional instances can be created• Microsoft Entra ID is the underlying product providing the identity service• The term <i>Tenant</i> means a single instance representing a single organization• The terms <i>Tenant</i> and <i>Directory</i> are often used interchangeably
<u>Azure subscription</u>	Used to pay for Azure cloud services

© Copyright Microsoft Corporation. All rights reserved.

Tenant is automatically created when I sign in any microsoft service in office.

Netflix example -> availing subscriptiion for user-> so creating tenant wont enable us to deploy machines -> we need subscription., to use any azure cloud services.

When you **sign in to a Microsoft service using a work or school account**, especially in an **organization like a company or educational institution**, Microsoft uses something called **Microsoft Entra ID** (formerly Azure Active Directory) to manage identities and access.

Here's what your trainer likely meant:

◊ What is a "Tenant" in Microsoft?

A **tenant** is like a dedicated space or container in Microsoft's cloud for your organization. It holds:

- User accounts
- Security settings
- Licenses
- Access to Microsoft services like Teams, Outlook, SharePoint, etc.

◊ How is it "automatically created"?

- If your organization **already has a Microsoft 365 or Azure subscription**, then the tenant already exists.
- But if someone from your organization **signs up for a Microsoft service for the first time** (like Microsoft 365 or Azure) using a work email, Microsoft may **automatically create a tenant** for that domain (e.g., @yourcompany.com).
- This tenant is then used to manage all users and services under that domain.

◊ Example:

Let's say your company email is arka@companyname.com. If no one from companyname.com has ever signed up for Microsoft services before, when you sign in, Microsoft might create a tenant for companyname.com and associate your account with it.

Would you like a simple diagram or visual to help explain this better? Or do you want to know how to check if your organization already has a tenant?

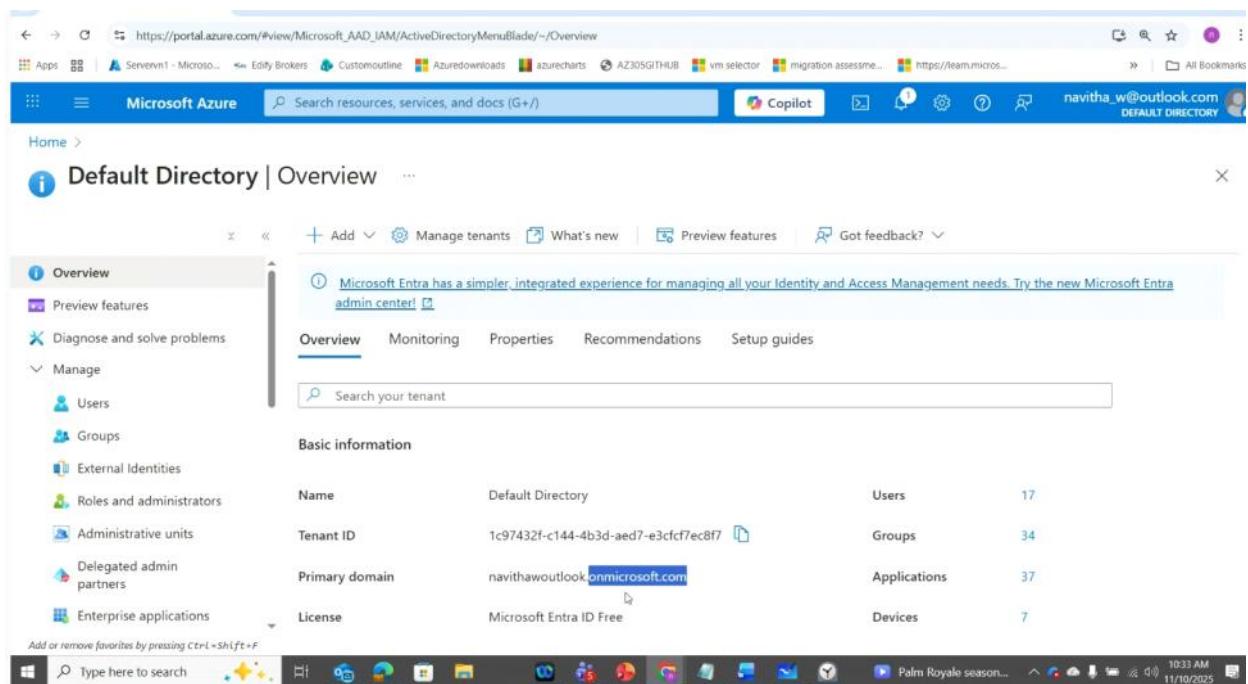
From <<https://www.microsoft365.com/chat?version=19.2511.34141.0&capabilities=interopPromise.suspendOnClose.autoStart&client-request-id=914931a4-e3b7-2346-b0b8-8c77b2fe8dce>>

Turn on your camera to select this layout.

Compare Microsoft Entra ID to Active Directory Domain Services

-  Microsoft Entra ID is primarily an identity solution
-  Queried using the REST API over HTTP and HTTPS
-  Uses HTTP and HTTPS protocols such as SAML, WS-Federation, and OpenID Connect for authentication (and OAuth for authorization)
-  Includes federation services, and many third-party services (such as Facebook)
-  Microsoft Entra ID users and groups are created in a flat structure, and there are no Organizational Units (OUs) or Group Policy Objects (GPOs)

© Copyright Microsoft Corporation. All rights reserved.



The screenshot shows the Microsoft Azure portal interface. The URL in the address bar is https://portal.azure.com/#view/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/~/Overview. The top navigation bar includes links for Apps, Serverv1 - Microsoft, Edgy Brokers, Customoutline, Azuredownloads, azurecharts, AZ305GITHUB, vm selector, migration assessment, https://team.microsoft.com, and All Bookmarks. The user's email, navitha.w@outlook.com, is displayed in the top right corner.

The main content area is titled "Default Directory | Overview". It features a sidebar with navigation links: Overview, Preview features, Diagnose and solve problems, Manage (with sub-options: Users, Groups, External Identities, Roles and administrators, Administrative units, Delegated admin partners, Enterprise applications), and a note about switching to the new Microsoft Entra admin center. The main panel displays basic information for the tenant, including:

Name	Default Directory	Users	17
Tenant ID	1c97432f-c144-4b3d-aed7-e3cfef7ec8f7	Groups	34
Primary domain	navithawoutlook.onmicrosoft.com	Applications	37
License	Microsoft Entra ID Free	Devices	7

At the bottom of the screen, the taskbar shows various pinned icons and the system tray indicates the date and time as 10:33 AM 11/10/2025.

SSO is a feature of entra id

What is self-service password reset in Microsoft Entra ID?

1. Determine who can use self-service password reset
2. Choose the number of authentication methods required and the methods available (email, phone, questions)
3. You can require users to register for SSPR (same process as MFA)

© Copyright Microsoft Corporation. All rights reserved.

What is self-service password reset in Microsoft Entra ID?

1. Determine who can use self-service password reset
2. Choose the number of authentication methods required and the methods available (email, phone, questions)
3. You can require users to register for SSPR (same process as MFA)

© Copyright Microsoft Corporation. All rights reserved.



P1 license required for SSPR

Global administrator -> highest privilege role for entra id

Entra id roles -> for entra id features only

Password admin also is another role.

Go to diff directory to see diff roles ..now we are concerned about only entra id roles

Microsoft Azure Search resources, services, and docs (G+)

navitha_w@outlook.com DEFAULT DIRECTORY

Home > Default Directory | Roles and administrators > Roles and administrators | All roles

All roles

New custom role Delete custom role Download assignments Refresh Preview features Got feedback?

To create custom roles, your organization needs Microsoft Entra ID Premium P1 or P2. Start a free trial.

Description	Privileges	Type	Actions
Authentication Policy Administrator		Built-in	...
Azure DevOps Administrator		Built-in	...
Azure Information Protection		Built-in	...
B2C IEF Keyset Administrator	PRIVILEGED	Built-in	...
B2C IEF Policy Administrator		Built-in	...
Billing Administrator		Built-in	...
Cloud App Security Administrator		Built-in	...
Cloud Application Administrator	PRIVILEGED	Built-in	...

Add or remove favorites by pressing Ctrl+Shift+F

10:54 AM 11/10/2025

Microsoft Azure Search resources, services, and docs (G+)

navitha_w@outlook.com DEFAULT DIRECTORY

Home > Default Directory | Roles and administrators > Roles and administrators | All roles > Global Administrator

Global Administrator | Description

All roles

Got feedback?

Diagnose and solve problems

Manage

Assignments

Description

Activity

Troubleshooting + Support

API Permission	Description
microsoft.directory/appConsent/appConsentRequests/allProperties/read	Read all properties of consent requests for applications registered with Microsoft Entra ID
microsoft.directory/applications/allProperties/allTasks	Create and delete applications, and read and update all properties
microsoft.directory/applications/synchronization/standard/read	Read provisioning settings associated with the application object
microsoft.directory/applicationTemplates/instantiate	Instantiate gallery applications from application templates
microsoft.directory/auditLogs/allProperties/read	Read all properties on audit logs, excluding custom security attributes audit logs
microsoft.directory/authorizationPolicy/allProperties/allTasks	Manage all aspects of authorization policy

Add or remove favorites by pressing Ctrl+Shift+F

10:56 AM 11/10/2025

The screenshot shows the Microsoft Entra admin center interface. The left sidebar is collapsed, showing options like Home, Agents, Favorites, and Entrada ID. The main content area is titled "Roles and administrators | All roles" under the "Contoso" tenant. It displays a list of roles with columns for Role, Description, Privileged, Assigned, Type, and Actions. The "AI Administrator" role is selected, showing its detailed description: "Manage all aspects of Microsoft 365 Copilot and AI-related enterprise services in Microsoft 365." Other roles listed include Application Administrator, Application Developer, Attack Payload Author, Attack Simulation Administrator, and Attribute Assignment Administrator.

Entra id is a service, not a role

Entra ID Roles (formerly Azure AD roles)

These are specific to identity management:

Role Name	What It Can Do
Global Administrator	Full control over Entra ID (can assign roles, reset passwords, etc.).
User Administrator	Can manage users and groups.
Security Administrator	Can manage security-related features.
Application Administrator	Can manage app registrations and permissions.

Not all users in one tenant

1 tenant -> Iti mindtree bangalore employees
Another tenant -> Iti kolkata

Tenants are given below:

Microsoft Azure Search resources, services, and docs (G+/)

Copilot

navitha.w@outlook.com DEFAULT DIRECTORY

Portal settings | Directories + subscriptions

Switching directories will reload the portal. The directory you choose will impact the subscription, resource group, and region filters that are available in the portal. Learn more about directories.

Directories + subscriptions

Appearance

Language + region

My information

Signing out + notifications

Useful links

- Learn more about settings
- Safelist URLs
- Microsoft partner network
- Privacy Statement
- More Azure resources
- Provide feedback

Current directory: Default Directory

Startup directory: Last visited (change)

Favorites All Directories

Search

Directory name ↑↓	Domain ↑↓	Directory ID ↑↓
azurenwtenant1	az30Sqege.onmicrosoft.com	d12fd449-9e08-4f8a-97ab-57a65f...
cloudaz305	cloud123ewqre.onmicrosoft.com	ae1d8033-48eb-44eb-bbe7-33a3...
contoso	azabc123se.onmicrosoft.com	739391c8-6a97-4715-9b47-d708a...
newcloud123	newcloud123.onmicrosoft.com	518f848b-1ad1-478c-951e-f95d5...
newext123q	newtxt123q.onmicrosoft.com	7b5ab1a9-a20d-4e36-a205-dbdb...
newnone	new123123.onmicrosoft.com	a89460d7-2ea4-4b8f-b841-f00c7...
newtcs123	newdemotcs.onmicrosoft.com	505a4c26-81f3-4b07-8e13-e49a5...

Type here to search

28°C Partly sunny 11:03 AM 11/10/2023

Creating Tenants:

Resource group and tenant diff

https://portal.azure.com/#view/Microsoft_AAD_IAM/DirectorySwitchBlade/subtitle/

Microsoft Azure Search resources, services, and docs (G+/)

Copilot

navitha.w@outlook.com DEFAULT DIRECTORY

Manage tenants

Home > Default Directory | Overview >

...

• • •

Microsoft Azure Search resources, services, and docs (G+)

Home > Default Directory | Overview > Manage tenants > Create a tenant ...

Microsoft Entra ID and Azure AD B2C enable users to access applications published by your organization, and share same administration experiences. [Learn more](#)

Tenant type

Customers must own a paid license to create Microsoft Entra Workforce tenant.

As of May 1 2025, Azure AD B2C is no longer available for new sales. If you are an existing customer, follow this link [Learn more](#)

Select a tenant type *

- Microsoft Entra ID
- Azure AD B2C
- Microsoft Entra External ID

[Help me choose](#)

Review + create < Previous Next : Configuration >

Creating tenants and switching it in azure portal settings
All tenants are isolated

Default Directory - Microsoft A https://portal.azure.com/#view/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/~/Overview

Microsoft Azure Search resources, services, and docs (G+)

Copilot

Home > **Default Directory | Overview** ...

Overview Add Manage tenants What's new Preview features Got feedback? ▾

Microsoft Entra has a simpler, integrated experience for managing all your Identity and Access Management needs. Try the new Microsoft Entra admin center! ↗

Overview Monitoring Properties Recommendations Setup guides

Search your tenant

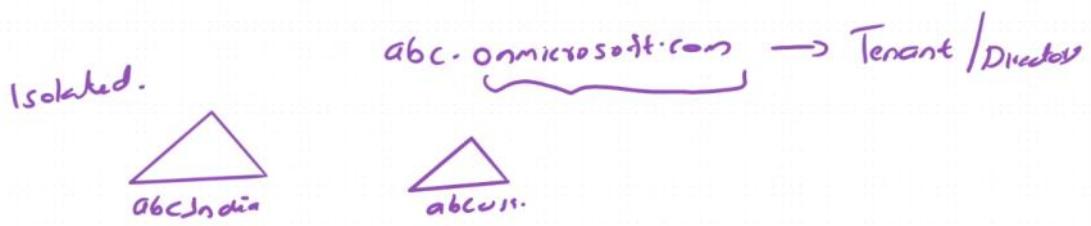
Basic information

Name	Default Directory	Users	17
Tenant ID	1c97432f-c144-4b3d-aed7-e3cfefec8f7	Groups	34
Primary domain	navithawoutlook.onmicrosoft.com	Applications	37
License	Microsoft Entra ID Free	Devices	7

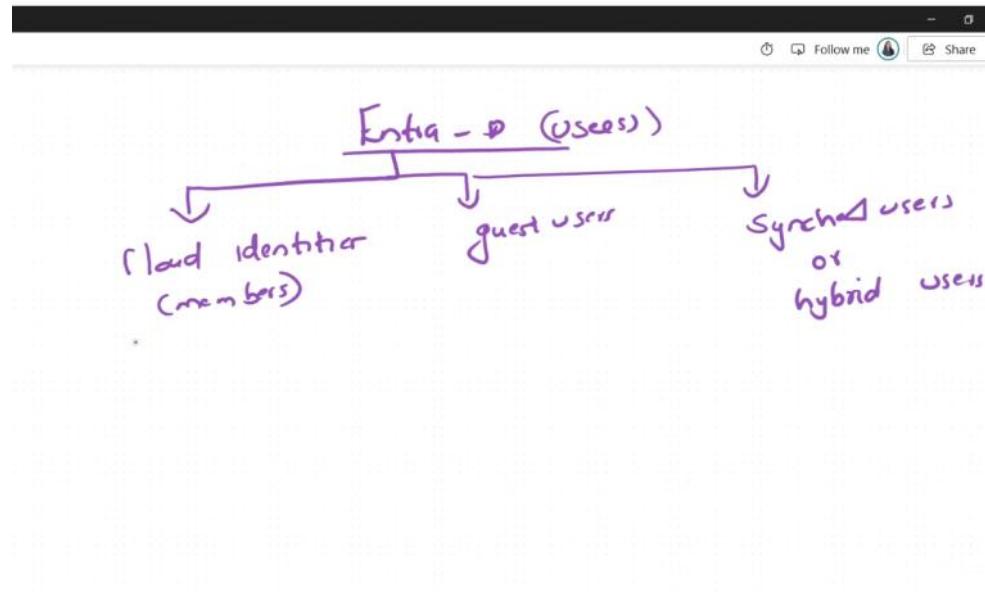
Add or remove favorites by pressing Ctrl+Shift+F

abc.onmicrosoft.com → Tenant / Director

Default tenant



Tenants created under my tenant. They are isolated, and authentication through entra -id.



Identities in cloud itself -> cloud identities

Cloud identity:

Microsoft Azure Search resources, services, and docs (G+/-) Copilot Home > Default Directory | Users > Users Default Directory

All users Audit logs Sign-in logs Diagnose and solve problems Deleted users Password reset User settings Bulk operation results Bulk operation results (Preview) New support request

Azure Active Directory is now Microsoft Entra ID.

Search Add filter 18 users found

Display name ↑	User principal name ↓	User type	Is Agent	On-premises sync	Identities
aakash	aakash@navithawout...	Member	No	No	navithawout...
Abdul Rahuman M S.	abdulrahumanms...	Guest	No	No	ExternalAzur...
aby	aby@navithawout...	Member	No	No	navithawout...
anurag	Up.anurag_gmail...	Guest	No	No	mail...
Mariyam Thomas	mariyamt_cloudb...	Guest	No	No	ExternalAzur...

Add or remove favorites by pressing Ctrl+Shift+F

Type here to search

Microsoft Azure Search resources, services, and docs (G+/-) Copilot Home > Default Directory | Users > Users > Create new user Create a new internal user in your organization

User principal name * aakash @ navithawoutlook.onmicrosoft.com ↗ Domain not listed? Learn more ↗

Mail nickname * aakash Derive from user principal name

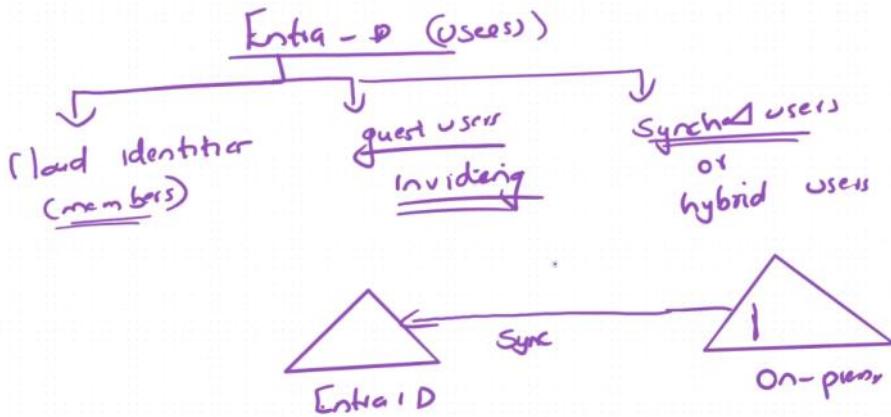
Display name * aakash

Password * Auto-generate password

Account enabled

Review + create < Previous Next: Properties > Give feedback

Type here to search



Entia-id connect /

Difference between entia-id and on-premise of LTIMIndtree

The screenshot shows the Microsoft Azure portal interface with the URL [https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-users-external-users#invite-an-external-user-to-collaborate-with-your-organization](#). The page title is "Invite external user".

The form fields are as follows:

- Email: sravan0808@gmail.com
- Display name: sravan
- Invitation message: hi, inviting to my tenant
- Send invite message:
- Message: (empty)
- Cc recipient: (empty)
- Invite redirect URL: <https://myapplications.microsoft.com/?tenantid=1c97432f-c144-4b3d-aed7-e3cfef7ec8f7>

EXT added to guest user

Guest user vs member

We are signing in a guest user we created Akash, and also changed password since I have admin privileges

Creating bulk users

Screenshot of the Microsoft Azure portal showing the 'Bulk create users' feature. The left sidebar shows 'Users' under 'Default Directory'. The main area displays a list of users with columns for 'Display name', 'User principal name', 'User type', and 'Is Active'. A warning message at the top right states: 'Please be aware of the bulk operations service limitations before using this feature. Operations can only run for up to 1 hour and has known issues in large tenants. Click here to learn more.' To the right, there's a step-by-step guide for bulk import:

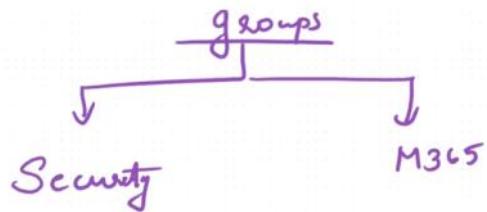
1. Download csv template (optional) [Download](#)
2. Edit your csv file
3. Upload your csv file

[Learn more about bulk import users](#)

Display name	User principal name	User type	Is Active
aakash	aakash@navithaw...	Member	No
Abdul Rahman M S.	abdulrahumanms...	Guest	No
aby	aby@navithawout...	Member	No
anurag	Up.anurag_gmail...	Guest	No
Mariyam Thomas	mariyamt_cloudth...	Guest	No
Miroslav Buchnice...	Miroslav.Buchnice...	Guest	No

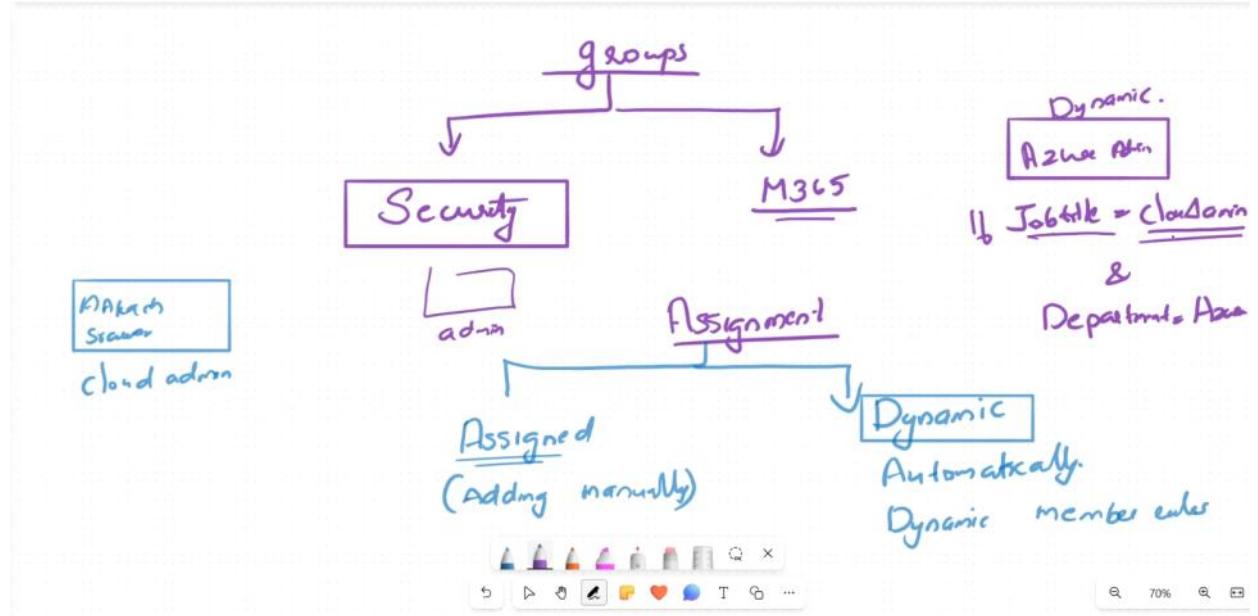
Groups

Security groups and m365 groups



Sg -> admin group, cloud-admin group, azure-admin group -> we can assign members
We can create dynamic groups -> automatically adding based on dynamic member rules

Suppose if job_title=cloud-admin -> all those members will be automatically added to our dynamic group azure-admin group. We are creating dynamic rules



P1 license at least for dynamic group

New Group

Got feedback?

Group type * Security

Group name * Itmindtree

Group description Itmindtree

Membership type Assigned

Owners
No owners selected

Create

Navitha Wilson (Unverified)

Lab 01 – Manage Microsoft Entra ID Identities



Job Skills

In this lab, you learn about users and groups.

Users and groups are the basic building blocks for an identity solution.

You create a new user and invite a guest user.

You also create a group and add a member and owner.

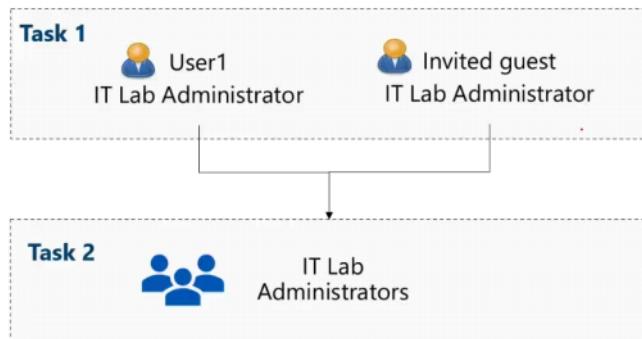
Task 1: Create and configure user accounts.

Task 2: Create groups and add members.

Next slide for an architecture diagram →

© Copyright Microsoft Corporation. All rights reserved.

Lab 01 – Manage Entra ID Identities (architecture diagram)



© Copyright Microsoft Corporation. All rights reserved.

Navitha Wilson (Unverified)

How would you like to sign in?

- Microsoft Account
- Entra ID
- Skillable Account

Welcome to the Microsoft Global Systems Integrator Portal hosted by Skillable.

Meeting chat

Some people in this chat are outside your org. It's p... they have message-related policies that will apply to chat. Learn more

Any user with login credential(username and Password) is called as identity.

Sayandeep Adhikary 10:12 AM

Premraj S 10:12 AM

PS kerberos

Pavithra Ramasamy 10:13 AM

PR in on Prem environment if we have third party IDP how Active directory will come into picture?

Sayandeep Adhikary 10:15 AM

PS Entra ID

You can't send messages because you are member of the chat.

Site Administration

Classes • Find Classes • Find Class Sessions	Class Enrollments • Find Enrollment Training Keys	Instructors • Create Instructor Blockout Time • Find Instructor Blockout Times
Material Vendors • Find Material Vendors	Materials • Find Materials • View Materials List	Courses • Find Courses

Navitha Wilson (Unverified)

Training key

2620193 - CloudThat - LTIMindtree x +

<https://gsi.learnondemand.net/Class/718571>

2620193 - CloudThat - LTIMindtree - AZ-104T00-A Microsoft Azure Administrator [Cloud Slice Provided]

Roster Pre-Class Checklist Sign-In Sheet Attendance Chart Manage Files Monitor Labs Training Keys

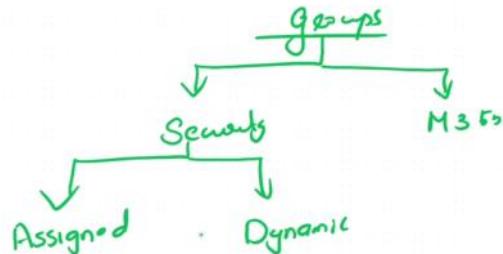
Basic Information

When: Monday, November 10, 2025 8:30 AM - Wednesday, November 19, 2025 2:30 PM (India Standard Time UTC05:30:00)
 Organization: Microsoft Global System Integrator (GSI)
 Post Class Lab Access Expiration Date: Monday, May 18, 2026
 Instructors: 1. Naveen H
 2. Navitha Wilson
 Instructor Phone Support: 1 (855) 438-5953
 Event Training Key: C0081FAA86D546F6
 MTM Survey Link:

Status

Scheduled
 Enrollments: 25/500
 This class has met the minimum enrollment requirement (0).

11.11.2025



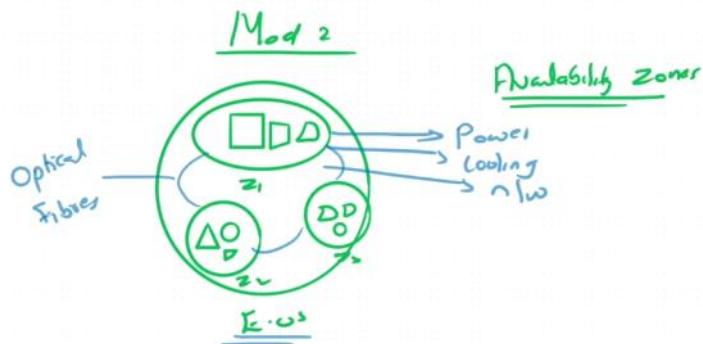
Module 2 Administrator Governance and compliance

RBAC Roles for rbac features just like Entra id roles for entra id features

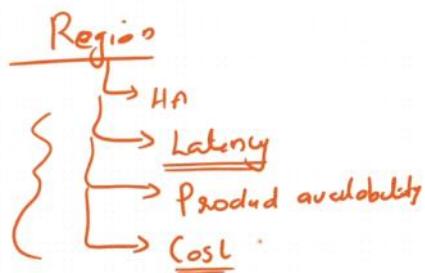
Hierarchy

Region

Availability zones -> group of datacentres.



Paired region -> us east then us west. -> regional pair if region got calamity



Container for billing -> azure subscription

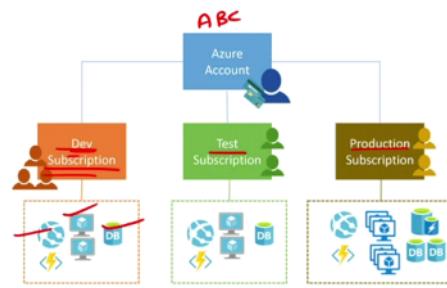
Implement Azure Subscriptions

↳ Container for billing

Only identities in Entra ID, or in a directory that is trusted by Entra ID, can create a subscription

Logical unit of Azure services that is linked to an Azure account

Security and billing boundary*



Subscription is entry point to azure resources

Identify Subscription Usage

Subscription	Usage
Free	Includes a \$200 credit for the first 30 days, free limited access for 12 months
Pay-As-You-Go	Charges you monthly
CSP	Agreement with possible discounts through a Microsoft Cloud Solutions Provider Partner – typically for small to medium businesses
Enterprise	One agreement, with discounts for new licenses and Software Assurance – targeted at enterprise-scale organizations
Student	Includes \$100 for 12 months – must verify student access

© Copyright Microsoft Corporation. All rights reserved.

Resource and resource group can be in different region. because rg is containing only metadata of resource.

Resource can be deployed only one resource group.

We can group resource group like for vms - one rg, for storage - one rf

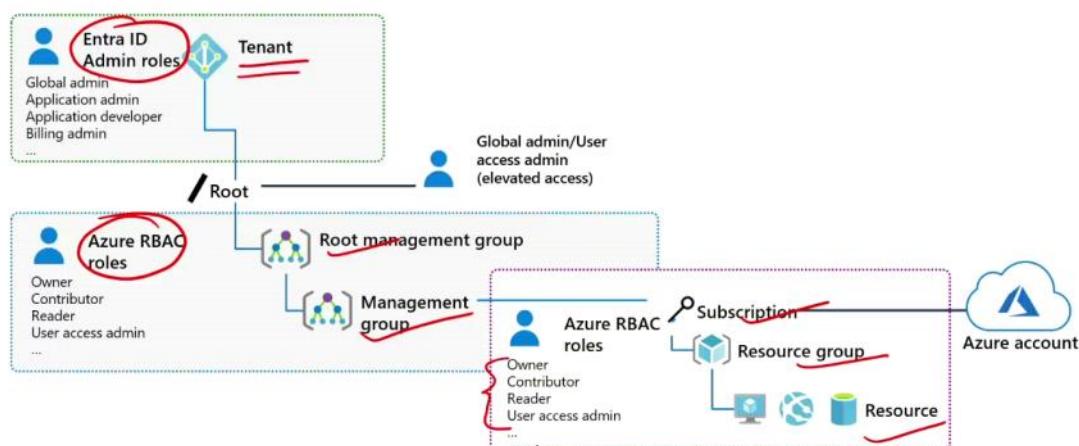
Renaming rg is not possible. rather move resource from one rg to another if needed.

if rg deleted -> all resource deleted. resource lock to avoid

locks -> delete and read only

can't do any operation on resource if read only.

Apply RBAC Authentication



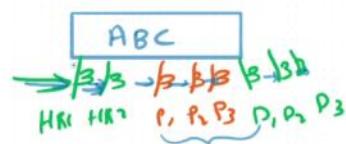
© Copyright Microsoft Corporation. All rights reserved.

Management Groups

I need to apply policy for all subscriptions

Management groups

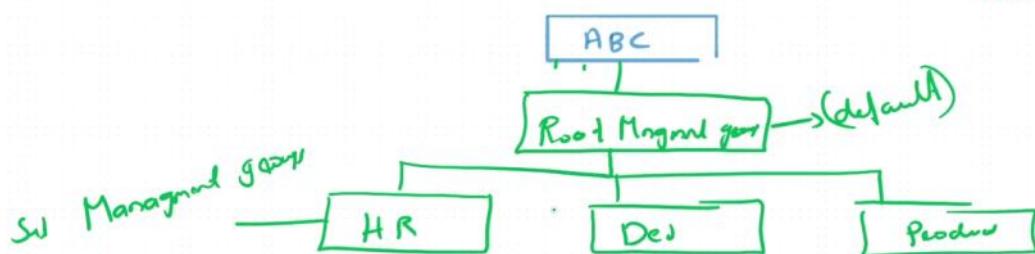
Rbac



Management groups can be used.

Management groups

Rbac



Create an Azure Resource Hierarchy

Management groups provides a level of scope above subscriptions

Target policies and spend budgets across subscriptions and inheritance down the hierarchies

Implement compliance and cost reporting by organization (business/teams)



* To prevent changes, apply resource locks at the subscription, resource group, or resources level

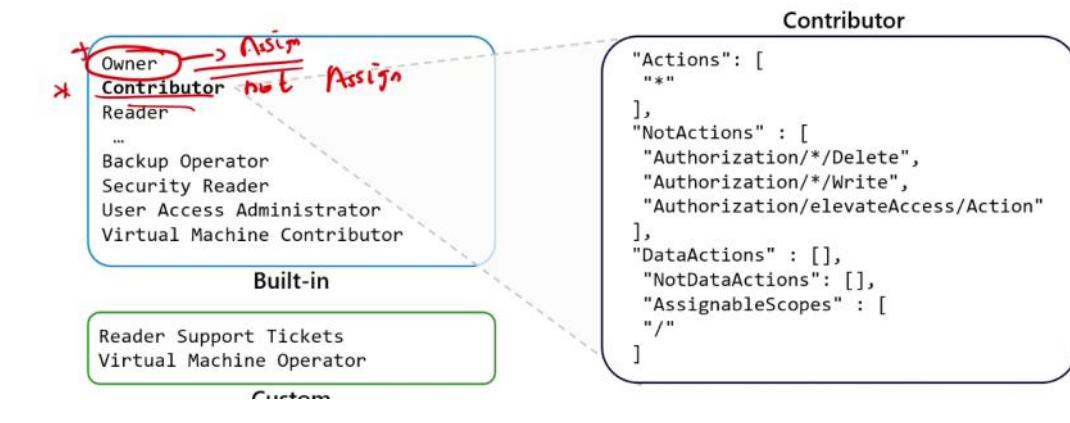
© Copyright Microsoft Corporation. All rights reserved.

Secure your Azure resources with Azure role-based access control (Azure RBAC)

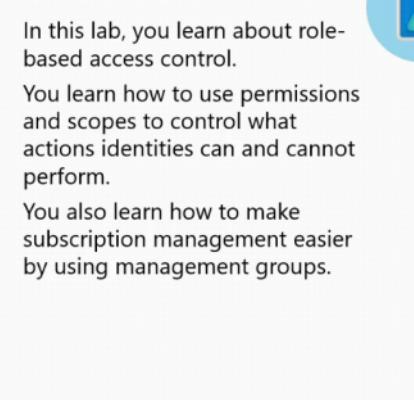
Na Wilson (Unverified)

Create a Role Definition

Collection of permissions that lists the operations that can be performed



Lab 02a – Manage Subscriptions and Azure RBAC



Job Skills

- Task 1: Implement management groups.
- Task 2: Review and assign a built-in Azure role.
- Task 3: Create a custom RBAC role.
- Task 4: Monitor role assignments with the Activity Log.

Lab 02b – Manage Governance via Azure Policy



Job Skills

In this lab, you learn how to implement your organization's governance plans.

You learn how Azure policies can ensure operational decisions are enforced across the organization.

You learn how to use resource tagging to improve reporting.

Task 1: Create and assign tags via the Azure portal.

Task 2: Enforce tagging via an Azure Policy.

Task 3: Apply tagging via an Azure Policy.

Task 4: Configure and test resource locks.

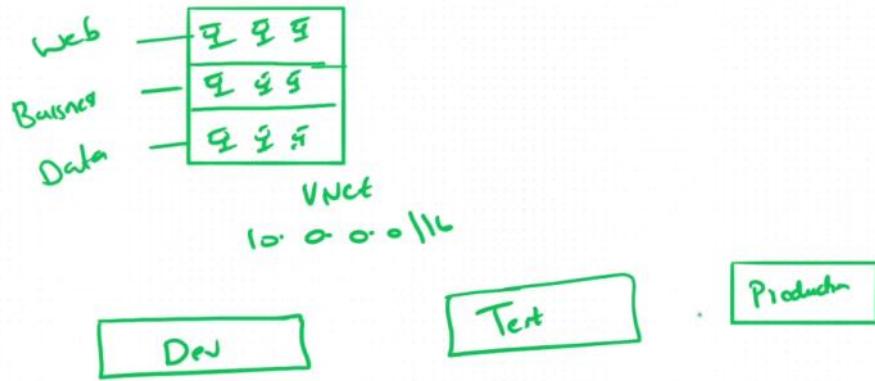
Next slide for an architecture diagram

© Copyright Microsoft Corporation. All rights reserved.

Deploy Azure resources with templates



© Copyright Microsoft Corporation. All rights reserved.



Code as a service
ARM Template

ARM template
Parameter
Variables
User defined function
Resources
Output

Microsoft Azure

Custom deployment

Select a template

Common templates

- Create a Linux virtual machine
- Create a Windows virtual machine
- Create a web app
- Create a SQL database
- Azure landing zone

Start with a quickstart template or template spec

Custom template for testing, prod and development

Parameter file

Deploy vm through custom template method

Review ARM Template Advantages

Improves consistency and promotes reuse

Reduce manual, error prone, and repetitive tasks

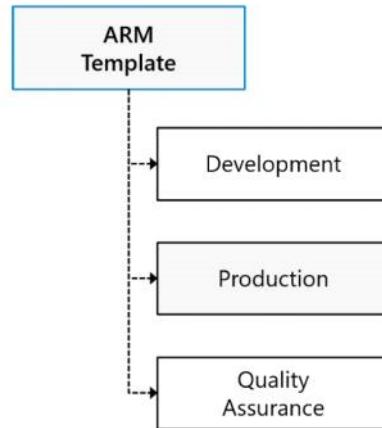
Express complex deployments

Express requirements through code

Provides validation tasks

Modular and can be linked

Simplifies orchestration



© Copyright Microsoft Corporation. All rights reserved.

Explore the JSON Template Schema

Defines all the Resource manager resources in a deployment

Written in JSON

A collection of key-value pairs

Each key is a string

Each value can be a string, number, Boolean expression, list of values, object

```
{  
    "$schema":  
        "http://schema.management.  
        azure.com/schemas/2019-04-  
        01/deploymentTemplate.json#",  
    "contentVersion": "",  
    "parameters": {},  
    "variables": {},  
    "functions": [],  
    "resources": [],  
    "outputs": {}  
}
```

© Copyright Microsoft Corporation. All rights reserved.

Consider Azure Bicep Files

Simpler syntax for writing templates

Smaller module files you can reference from a main template

Automatically detect dependencies between your resources

Visual Studio Code extension with validation and IntelliSense

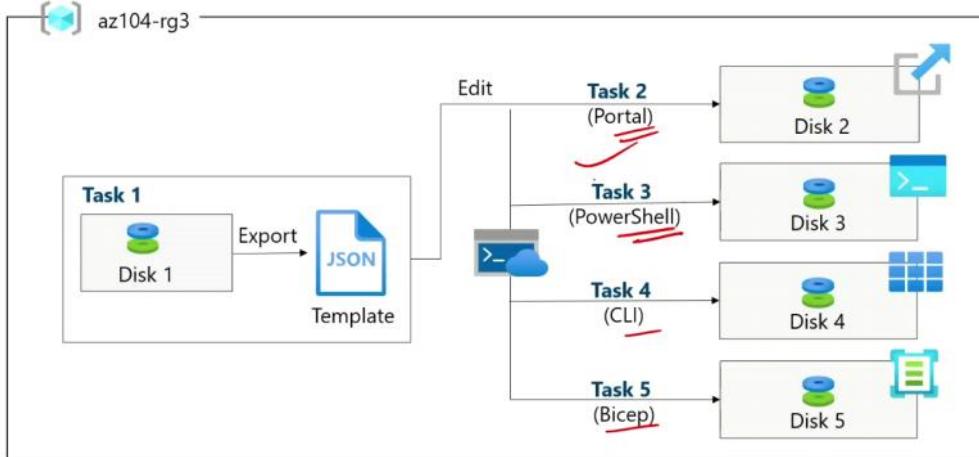
Bicep file

```
resource storageAccount 'Microsoft.Storage/storageAccounts@2021-01-01' = {
  name: storageAccountName
  location: location
  tags: {
    displayName: storageAccountName
  }
  kind: 'StorageV2'
  sku: {
    name: 'Standard_LRS'
  }
}
```



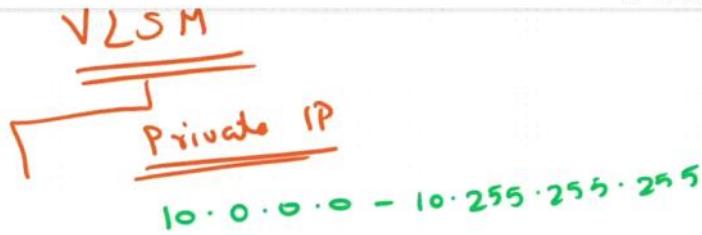
© Copyright Microsoft Corporation. All rights reserved.

Lab 03 – Architecture diagram



© Copyright Microsoft Corporation. All rights reserved.

Subnet masking
VLSM



/24 notation

Borrowing bits from host \rightarrow /25 \rightarrow VLSM \rightarrow to avoid wastage

\therefore

$192 \cdot 168 \cdot 10 \cdot 0 \underline{\underline{25}}$

$\begin{array}{l} \frac{3}{2} = 2 = \underline{\underline{2}} \text{ Network} \\ \frac{9}{2} = 2 - \text{Host} = \underline{\underline{128}} \end{array}$

$\begin{array}{l} |25 \rightarrow 128 \\ |26 \rightarrow 64 \\ |27 \rightarrow 32 \\ |28 \rightarrow 16 \\ |29 \rightarrow 8 \\ |30 \rightarrow 4 \\ |31 \rightarrow 2 \end{array}$

The screenshot shows the CIDR Calculator interface. At the top, there's a search bar with the URL https://cidr.xyz and a navigation bar with links like CIDR Calculator, Subnet Guide, Embed, and Github. The main title is "CIDR Calculator" with the subtitle "Visualize and Calculate Network Ranges with IPv4 CIDR Blocks". Below this, the IP address 192.168.10.144 is shown with a slash followed by the subnet mask length 28. The binary representation of the IP and subnet mask is displayed below. The calculated values are:

- Netmask:** 255.255.255.240
- CIDR Base IP:** 192.168.10.144
- Broadcast IP:** 192.168.10.159
- Count:** 16
- First Usable IP:** 192.168.10.145
- Last Usable IP:** 192.168.10.158

The taskbar at the bottom shows various pinned icons and the user's name Navitha Wilson (Unverified).

Management groups are used to logically organize and segment subscriptions.

For example, if your organization has a dedicated support team for Europe, you can organize European subscriptions into a management group to provide the support staff access to those subscriptions (without providing individual access to all subscriptions).

Azure Vnet -> software defined.

regional service
pvt

NSG -> allows Subnet to exclude connecting with the other subnet in vnet.

Great question! **Application Security Groups (ASGs) are applied at the NIC (Network Interface Card) level**, not at the subnet level.

Here's why:

- Each VM in Azure has a NIC, and that NIC is what connects the VM to the virtual network.
- When you assign an ASG, you associate it with the NIC of the VM, meaning the VM becomes part of that ASG.
- NSG rules then reference ASGs as the source or destination for traffic filtering.

Key Points

- **ASG ≠ Subnet:** ASGs do not apply to entire subnets; they apply to individual NICs.
- You can have multiple ASGs on a single NIC.
- This gives **fine-grained control** over which VMs belong to which security group, even if they share the same subnet.

Example:

- Subnet: 10.0.0.0/24
- VMs: VM1, VM2, VM3
- Assign ASG WebServers to NIC of VM1 and VM2.
- Assign ASG DBServers to NIC of VM3.
- NSG rule: Allow traffic from WebServers to DBServers on port 1433.

Do you want me to **create a visual diagram** showing how ASG attaches to NIC and how NSG rules reference ASGs? Or should I give you a **step-by-step Azure Portal setup guide?**