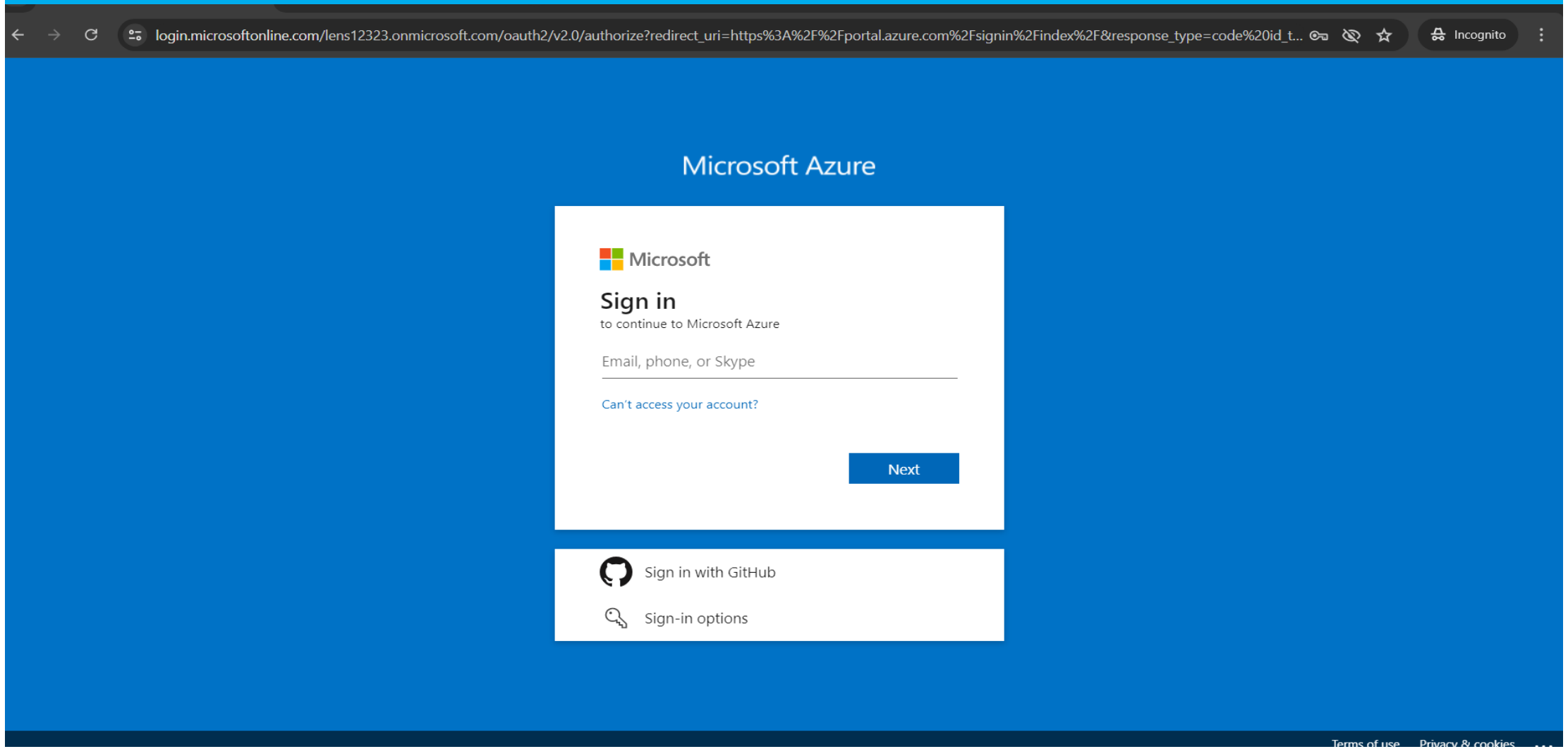
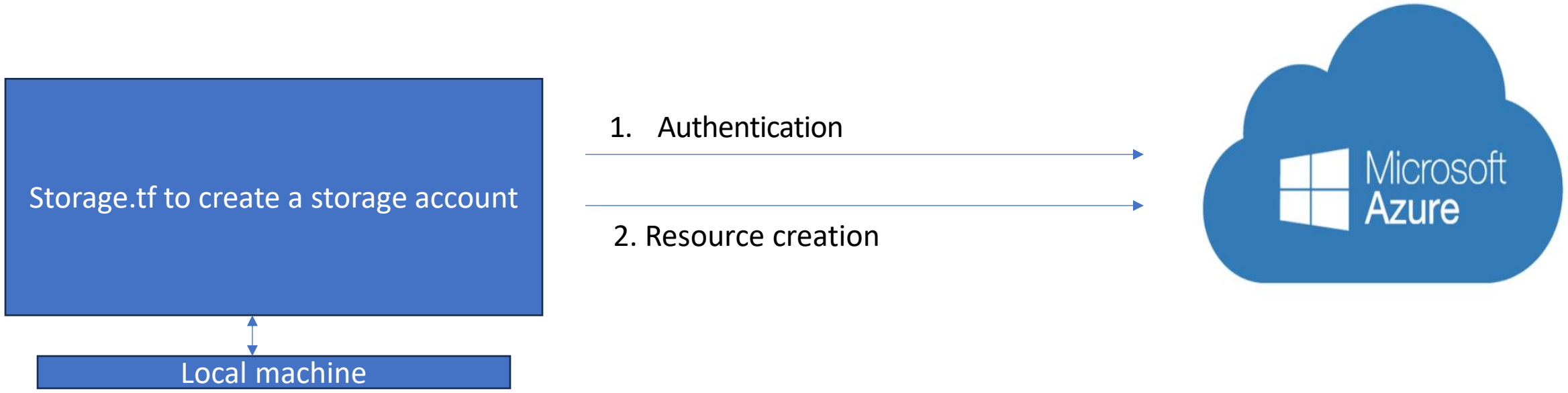
Authentication with Terraform

Manual Process Authentication with Azure



Authentication with Terraform

Before diving into managing environments or creating a resources with Terraform, the crucial initial step involves Authentication and Authorization

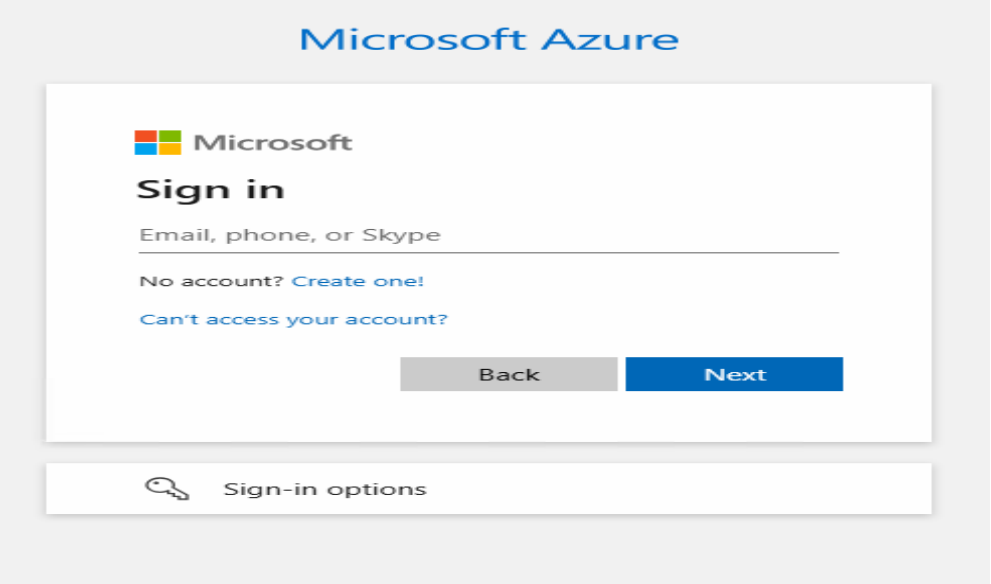


Authentication with Terraform

First Step : open a CMD & Type AZ login

```
PS C:\Users\sanu> az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web browser. If no web browser is available or if the web browser fails
to open, use device code flow with `az login --use-device-code`.
```

Second Step: A web browser will be opened to enter the username & password for the authentication.



Authentication with Terraform

Third Step :Once you authenticate yourself, the below message will come.

```
C:\Users\sanu>az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web browser. If no web browser is available or if the web browser fails
to open, use device code flow with `az login --use-device-code`.

Retrieving tenants and subscriptions for the selection...
The following tenants don't contain accessible subscriptions. Use `az login --allow-no-subscriptions` to have tenant level access.
298a5fec-5285-4b16-a565-6ef1144d7707 'Default Directory'

[Tenant and subscription selection]

No     Subscription name    Subscription ID                       Tenant
-----  -------------------  ------------------------------------  --------
[1] *  new                  8512a432-b466-4c66-8947-7f750e26c92d  check12

The default is marked with an *; the default tenant is 'check12' and subscription is 'new' (8512a432-b466-4c66-8947-7f750e26c92d).

Select a subscription and tenant (Type a number or Enter for no changes):

Tenant: check12
Subscription: new (8512a432-b466-4c66-8947-7f750e26c92d)

[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration at https://go.microsoft.com/fwlink/?linkid=2271236

If you encounter any problem, please open an issue at https://aka.ms/azclibug

[Warning] The login output has been updated. Please be aware that it no longer displays the full list of available subscriptions by default.
```

Visual Studio Code Extensions

Azure CLI

- Extensions in Visual Studio enable you to customize and enhance your experience by adding new features or integrating existing tools.
- They provide a diverse array of functionalities including color management, auto-complete enhancements, and spell-checking capabilities
- HashiCorp also provides extension for Terraform for Visual Studio Code.

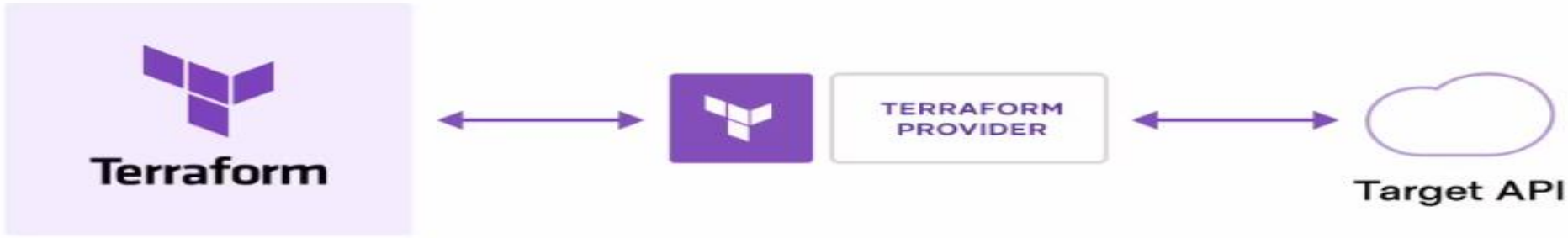
Providers in Terraform

- Terraform supports multiple vendors, and each vendor is represented by its own provider.

- Depending on what type of infrastructure we want to launch, we have to use appropriate providers . Eg: Azure task use Azure providers , Aws task use Aws providers.

How does Terraform provider works ?

Terraform creates and manages resources on cloud platforms and other services through their application programming interfaces (APIs). Providers enable Terraform to work with virtually any platform or service with an accessible API.

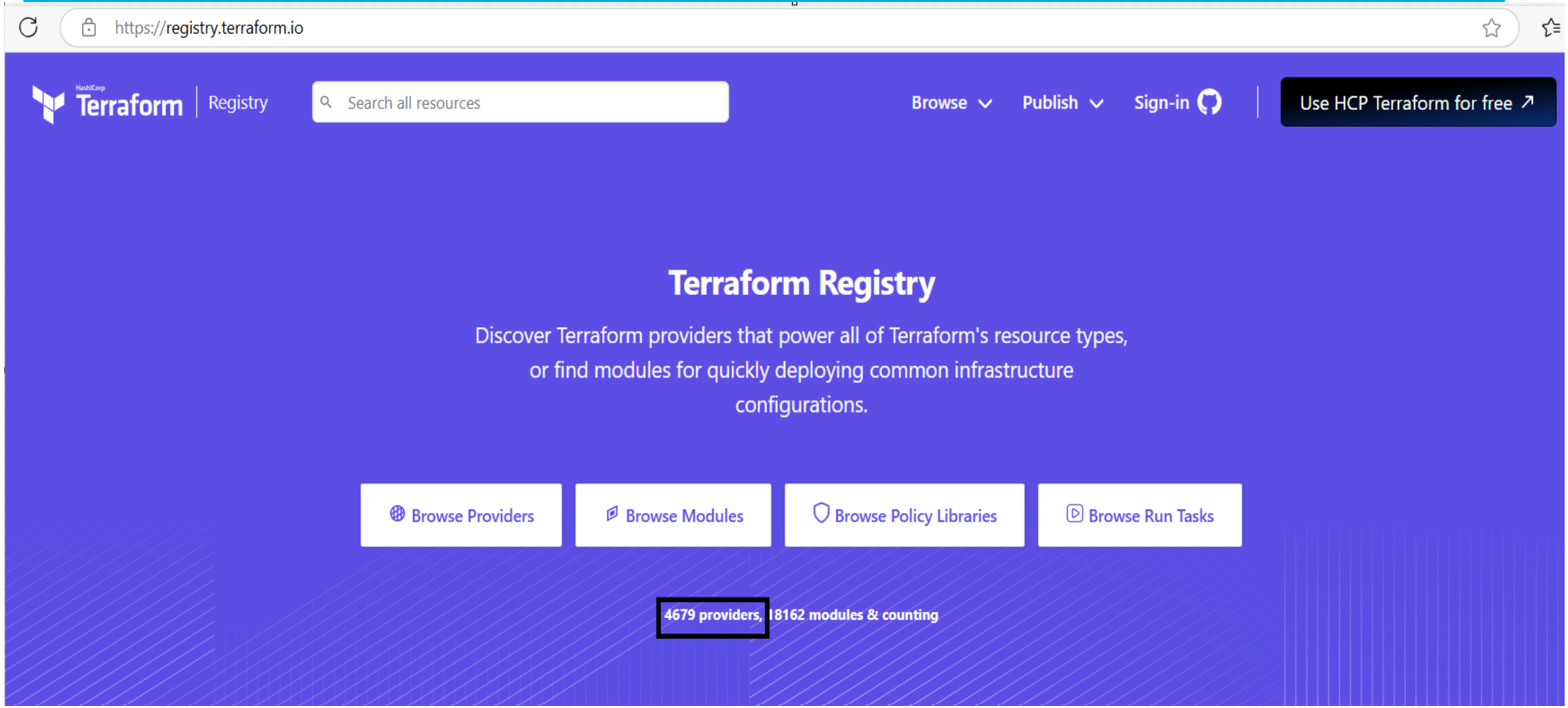


Important Question

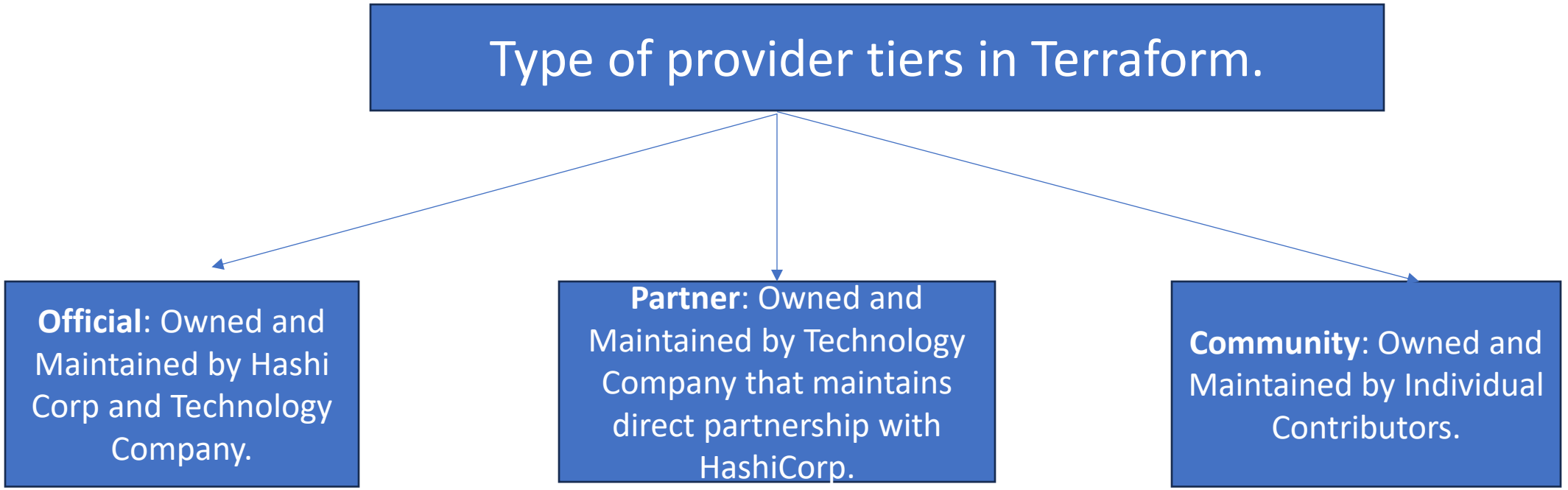
Today, I am working in Azure and learning Terraform through Azure. If in the future I start working in AWS, will I need to learn Terraform again

The core concepts and standard syntax remain similar across all providers. If you learn Terraform on one provider, you will be able to work with all providers easily

registry.terraform.io



Provider Maintainers



azurerm

Official by: HashiCorp

Public Cloud

Lifecycle management of Microsoft Azure using the Azure Resource Manager APIs. maintained by the Azure team at Microsoft and the Terraform team at HashiCorp

VERSION	PUBLISHED	SOURCE CODE
3.108.0	6 days ago	hashicorp/terraform-provider-azurerm

oci

Partner by:oracle

Infrastructure (IaaS)

Interact with the many resources supported by the Oracle Cloud Infrastructure via the OCI provider APIs.

VERSION	PUBLISHED	SOURCE CODE
5.46.0	9 days ago	oracle/terraform-provider-oci

adyen

by:weavedev

Utility

VERSION	PUBLISHED	SOURCE CODE
0.0.2	13 days ago	weavedev/terraform-provider-adyen

Use Provider

Providers / hashicorp / azurerm / Version 3.108.0 Latest Version

azurerm

Overview Documentation USE PROVIDER

azurerm

Official by: HashiCorp

Public Cloud

Lifecycle management of Microsoft Azure using the Azure Resource Manager APIs. maintained by the Azure team at Microsoft and the Terraform team at HashiCorp

VERSION	PUBLISHED	SOURCE CODE
3.108.0	6 days ago	hashicorp/terraform-provider-azurerm

How to use this provider

To install this provider, copy and paste this code into your Terraform configuration. Then, run `terraform init`.

Terraform 0.13+

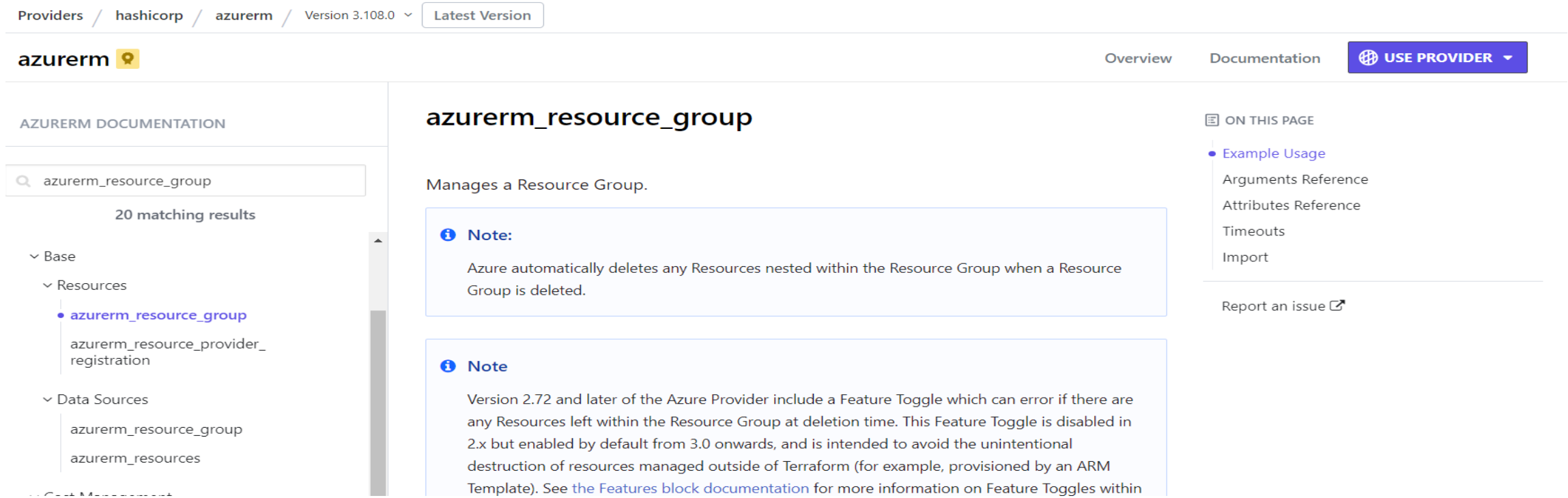
```
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "3.108.0"
    }
  }
}

provider "azurerm" {
  # Configuration options
}
```

Providers Demo

Resources in Terraform

Resources in Terraform

In Terraform, **resources** are the fundamental building blocks that describe infrastructure objects within your configuration. Each **resource block** represents one or more components, such as virtual networks, compute instances, or DNS records. These blocks define the desired state of various infrastructure elements, allowing you to manage and provision them effectively



Resources Demo

Join us in our Adventure

https://www.linkedin.com/in/akash-kumar-480b3858/

https://www.instagram.com/akash_sinha08/

Join us in our Adventure

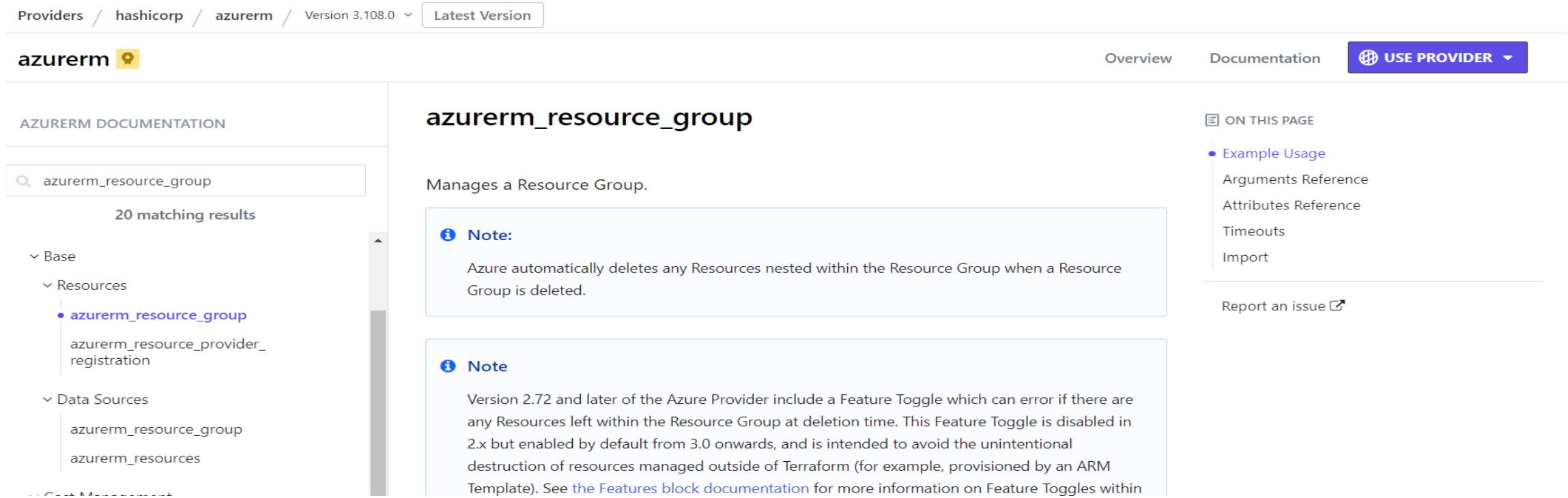
https://www.linkedin.com/in/akash-kumar-480b3858/

https://www.instagram.com/akash_sinha08/

Resources in Terraform

Resources in Terraform

In Terraform, **resources** are the fundamental building blocks that describe infrastructure objects within your configuration. Each **resource block** represents one or more components, such as virtual networks, compute instances, or DNS records. These blocks define the desired state of various infrastructure elements, allowing you to manage and provision them effectively



Resources Demo

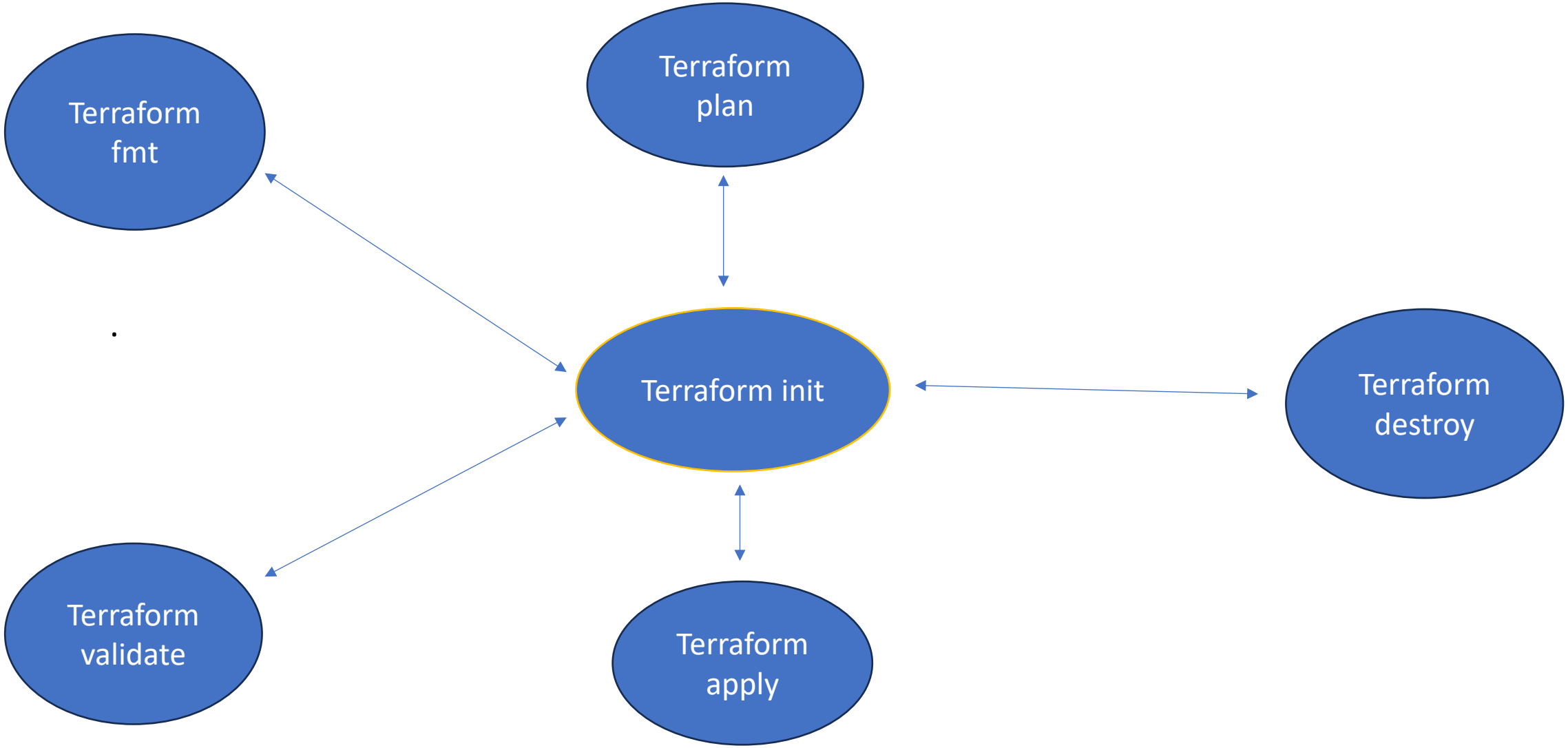
Join us in our Adventure

https://www.linkedin.com/in/akash-kumar-480b3858/

https://www.instagram.com/akash_sinha08/

Terraform Commands

Terraform Commands



Overview of Terraform Init

Initialization Phase

- Initializes a new or existing Terraform configuration.
- Downloads necessary providers and modules specified in the configuration.
- Creates a .terraform directory to store configuration and provider information locally

```
C:\Users\gaura\Desktop\Terraform_Demos>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/azurerm...
- Installing hashicorp/azurerm v2.65.0...
- Installed hashicorp/azurerm v2.65.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Overview of Terraform fmt

```
resource "azurerm_resource_group" "rg" {
  name = rg1
    location = "Central US"
}
```


After fmt

```
resource "azurerm_resource_group" "rg" {
  name     = rg1
  location = "Central US"
}
```

Before fmt

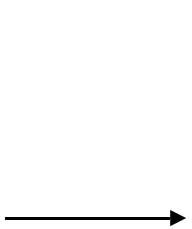
- The terraform fmt command is used to rewrite Terraform configuration files to take care of the overall formatting.

Overview of Terraform Validate

Terraform validate mainly checks if the configuration syntax is correct

It can check various aspects including unsupported arguments, undeclared variables and others.

```
resource "azurerm_resource_group" "rg" {
  name     = "rg1"
  locations = "Central US"
}
```



```
The argument "location" is required, but no definition was found.


Error: Unsupported argument

  on first_vm.tf line 7, in resource "azurerm_resource_group" "rg":
   7:   locations = "Central US"

An argument named "locations" is not expected here. Did you mean "location"?
```

Terraform Plan

The terraform plan command creates an execution plan. By default, creating a plan consists of:

- Analyzes and evaluates the Terraform configuration.
- Displays a preview of changes to be applied.
- Identifies new resources, modifications, and deletions.

```
C:\Users\gaura\Desktop\Terraform_Demos>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.example will be created
  + resource "azurerm_resource_group" "example" {
      + id       = (known after apply)
      + location = "westus"
      + name     = "testgroup1"
    }

Plan: 1 to add, 0 to change, 0 to destroy.
```

Terraform Apply

- Executes the planned changes after confirmation.
- Applies modifications to the infrastructure.
- Creates, updates, or deletes resources as per the configuration.

```
C:\Users\gaura\Desktop\Terraform_Demos>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.example will be created
  + resource "azurerm_resource_group" "example" {
      + id       = (known after apply)
      + location = "westus"
      + name     = "testgroup1"
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

Destroying Infrastructure with Terraform

- **terraform destroy** allows us to destroy all the resources that are created within the folder.

Approach 1

- **#terraform destroy**

Approach 2

- **terraform destroy** with **–target** flag allows us to destroy specific resource.
- **#terraform destroy –target azurerm_resource_group.example1**

```
resource "azurerm_resource_group" "example1" {
  name = "testgroup1"
  location = "West US"
}
```

```
resource "azurerm_resource_group" "example1" {
  name = "testgroup1"
  location = "West US"
}
```

```
resource "azurerm_resource_group" "example2" {
  name = "testgroup2"
  location = "West US"
}
```

Resource Type/Global Resource Name	Local Resource Name
azurerm_resource_group	example1
azurerm_resource_group	example2

Terraform Destroy with Target

- The –target option can be used to focus Terraform attention on only a subset of resources.
- **Combination of: Resource Type/Global Resource Name + Local Resource Name**

Join us in our Adventure

https://www.linkedin.com/in/akash-kumar-480b3858/

https://www.instagram.com/akash_sinha08/