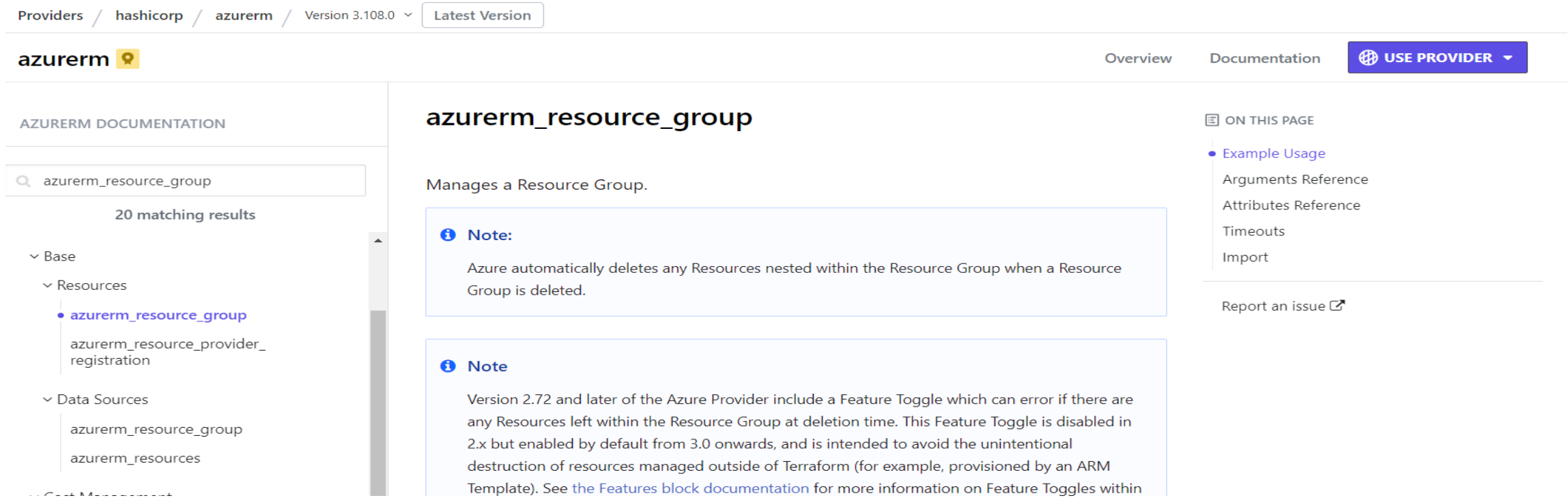
Resources in Terraform

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

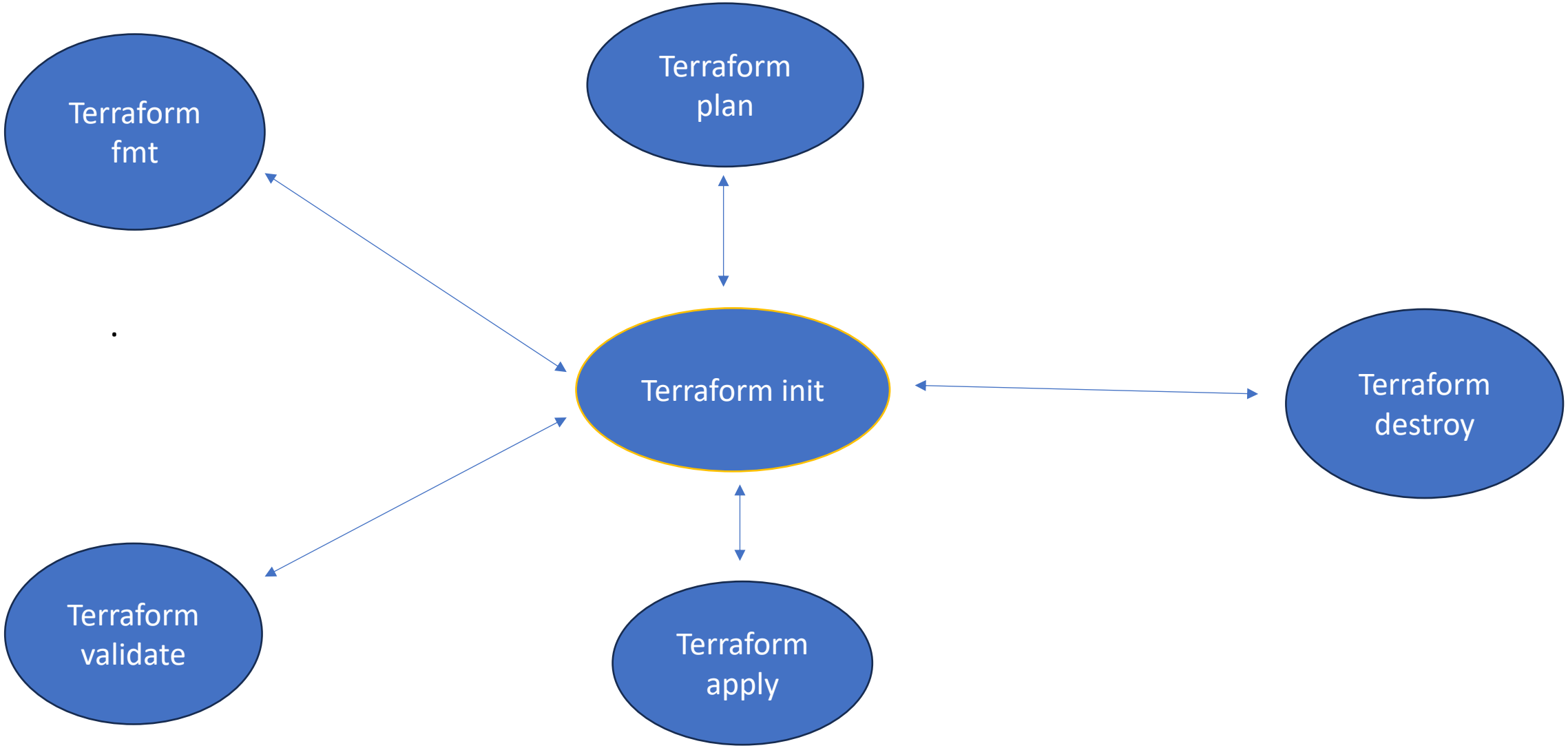
Join us in our Adventure

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/