**Terraform State File Management.**

1. **Delete specific resource from terraform**
   1. Delete or remove the resource block from the configuration file and then run **terraform plan and terraform apply**. This will remove the resource.
   2. **terraform destroy -target="azurerm\_subnet.tfsubnet3"**

// after that Delete or remove the resource block from the configuration file

1. **If a resources is delete from portal which was created by TF , then how to manage**

1. **Terraform state list** // it will show the list of resources exists in state file
2. **Terraform refresh** // it will remove the deleted resources from state file, however this command is depreciated. the refresh operation happens automatically with **terraform plan** or **terraform apply**. However, you can still force a refresh using the command.
3. However **terraform plan or apply** will recreate the deleted resource. If we want the delete resources as deleted then we need to remove resources block from configuration file.

1. **If a resource is created in portal directly , not in terraform then how to manage**

1. **When resource block is written in configuration file for the same resource which is created in portal directly. This error will come.**

Error: A resource with the ID "/subscriptions/66163d79-aaee-4101-a413-a8224073123a/resourceGroups/myrg700/providers/Microsoft.Network/virtualNetworks/vnet0999/subnets/snet066666" already exists - to be managed via Terraform this resource needs to be imported into the State. Please see the resource documentation for "azurerm\_subnet" for more information.

* **Terraform import <local name of resource block> <resource id from portal of the created resource>**

* **terraform import azurerm\_subnet.tfsubnet4 /subscriptions/66163d79-aaee-4101-a413-a8224073123a/resourceGroups/myrg700/providers/Microsoft.Network/virtualNetworks/vnet0999/subnets/snet066666**

**// this will modify the terraform state file with the stanza of resources created directly in portal**

1. **When resource block is not written in configuration file for the same resource which is created in portal directly. This error will come.**

Terraform does not automatically create the **resource** block in your configuration file. You need to generate the **resource block** that matches the state. Write a **Terraform resource block** in your configuration file that corresponds to the **resource** you created. Make sure the name and type match the **Azure resource**. And then do the **terraform import**

1. **How to manage when .tfstate file deleted**

If your Terraform state file (terraform.tfstate) is deleted or lost, you lose track of the current state of your infrastructure, which can lead to serious issues like resource drift, unexpected resource deletion, or duplication. Here's how you can manage and recover from this situation:

**1. Restore from Backup:**

* **Check for Backups:** Terraform often stores backup files with a .backup extension in the same directory as your terraform.tfstate file. Look for a file like terraform.tfstate.backup.
* **Restore the Backup:** If you find the backup file, rename it to terraform.tfstate and place it back in the appropriate directory.
* **Reinitialize Terraform:** After restoring, reinitialize Terraform:  
  **terraform init**

**2. Use Remote State (If Configured):**

If you were using a remote backend (like S3 for AWS, Azure Storage, or Terraform Cloud), you can recover the state from the remote location.

* **Reinitialize Backend:** Run terraform init to reinitialize Terraform and reconnect to the remote backend. Terraform will download the latest state from the remote backend.
* **Check for State in Remote Backend:** You can manually check the remote backend (e.g., S3 bucket, Azure Blob Storage) to see if the state file exists and is intact.

**3. Rebuild the State File Manually:**

If you cannot recover the state file from a backup or remote backend, you'll need to manually recreate the state file using terraform import. This is a more complex process but can be done as follows:

1. **Recreate Resources in Terraform Configuration:** Ensure that your Terraform configuration files (.tf files) match the current state of your infrastructure. You may need to update or recreate them based on what currently exists in your cloud environment.
2. **Import Existing Resources:** Use terraform import to bring existing resources back under Terraform management. For each resource:  
     
   > terraform import <resource\_type>.<resource\_name> <resource\_id>  
   > terraform import azurerm\_resource\_group.example /subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/myResourceGroup

1. **Repeat for All Resources:** Repeat the import process for all resources in your configuration. This will create a new state file that reflects your existing infrastructure.

**4. Plan and Validate:**

After importing all resources, run a terraform plan to ensure Terraform recognizes the current state correctly and does not try to recreate or destroy any resources.

**terraform plan and apply**