### **1. What is the difference between resource and data in Terraform, and how are they used?**

* **resource**: Used to **create, manage, and modify** infrastructure components (e.g., EC2 instances, S3 buckets).
* **data**: Used to **fetch existing resources** from the infrastructure without creating new ones.

**Example**:  
# Creating a new EC2 instance

resource "aws\_instance" "example" {

ami = "ami-12345678"

instance\_type = "t2.micro"

}

# Fetching an existing VPC

data "aws\_vpc" "existing\_vpc" {

id = "vpc-abcdef123"

}

**2. What are the most commonly used 10 Terraform commands and their use cases?**

| **Command** | **Use Case** |
| --- | --- |
| terraform init | Initializes Terraform in the working directory. |
| terraform plan | Shows what changes Terraform will make before applying them. |
| terraform apply | Applies the changes and provisions resources. |
| terraform destroy | Destroys all resources managed by Terraform. |
| terraform validate | Checks the configuration for syntax errors. |
| terraform fmt | Formats the Terraform code according to best practices. |
| terraform state list | Lists all resources tracked in the Terraform state file. |
| terraform state show <resource> | Shows details of a specific resource in the state file. |
| terraform output | Displays the output variables defined in Terraform. |
| terraform taint <resource> | Marks a resource for recreation on the next apply. |

### **3. What are the data types in Terraform?**

* **Primitive Types**:
  + string
  + number
  + bool
* **Complex Types**:
  + list(type)
  + map(type)
  + set(type)
  + object({})
  + tuple([])

Example:

variable "example\_list" {

type = list(string)

default = ["AWS", "Terraform"]

}

### **4. What is a function in Terraform?**

Functions in Terraform are used for string manipulation, type conversion, and arithmetic operations.

Example:

variable "name" {

default = "Terraform"

}

output "uppercase\_name" {

value = upper(var.name)

}

Common functions:

* length()
* upper()
* lower()
* join()
* split()
* lookup()

### **5. What are Terraform modules, and how do they function in an environment?**

* Terraform **modules** are reusable components that help **organize infrastructure**.
* Modules allow you to **group resources together** and use them across different projects.

Example:

module "ec2\_instance" {

source = "./modules/ec2"

instance\_type = "t2.micro"

}

Inside modules/ec2/main.tf:

resource "aws\_instance" "example" {

ami = "ami-12345678"

instance\_type = var.instance\_type

}

### **6. What is the Terraform lifecycle, and how does it work?**

Terraform follows this lifecycle:

1. **Write**: Define infrastructure in .tf files.
2. **Init**: Run terraform init to initialize providers.
3. **Plan**: Run terraform plan to preview changes.
4. **Apply**: Run terraform apply to create/update resources.
5. **Destroy**: Run terraform destroy to remove infrastructure.

### **7. If the Terraform state file is deleted but existing resources need to be managed without recreation, how can we achieve this?**

* **Solution**: Use terraform import to reintroduce existing resources into the state file.

terraform import aws\_instance.example i-1234567890abcdef

### **8. If the Terraform state file is deleted, can we recreate or modify existing resources? If so, how?**

Yes, by **importing** existing resources into Terraform state using:

terraform import <resource\_type>.<resource\_name> <resource\_id>

Example:

terraform import aws\_s3\_bucket.mybucket my-existing-bucket

### **9. How can multiple EC2 instances be created in Terraform without using the count parameter?**

We can use **for\_each**:

variable "instances" {

type = map(string)

default = {

"instance1" = "t2.micro"

"instance2" = "t2.small"

}

}

resource "aws\_instance" "example" {

for\_each = var.instances

ami = "ami-12345678"

instance\_type = each.value

}

### **10. What are local and variable definitions in Terraform?**

* **Variables**: Defined using variable blocks.
* **Locals**: Used for temporary values within a module.

Example:

variable "region" {

default = "us-west-2"

}

locals {

env = "dev"

}

### **11. If a child module is used for VPC and EC2 instances in Terraform, how can we deploy the EC2 instance in the VPC’s subnet?**

* Pass **VPC subnet ID** as an output from the VPC module and use it in the EC2 module.

Example:

modules/vpc/outputs.tf:

output "subnet\_id" {

value = aws\_subnet.main.id

}

In **EC2 module**:

resource "aws\_instance" "example" {

ami = "ami-12345678"

instance\_type = "t2.micro"

subnet\_id = var.subnet\_id

}

Then, call both modules:

module "vpc" {

source = "./modules/vpc"

}

module "ec2" {

source = "./modules/ec2"

subnet\_id = module.vpc.subnet\_id

}

### **12. What is the difference between a resource block and a data block in Terraform?**

* **Resource Block**: Used to **create and manage resources**.
* **Data Block**: Used to **fetch existing resources**.

Example:

resource "aws\_instance" "new\_ec2" {

ami = "ami-12345678"

instance\_type = "t2.micro"

}

data "aws\_vpc" "existing\_vpc" {

id = "vpc-abcdef123"

}