**Understanding Resource Dependencies in Terraform**

**Implicit Dependency**

. By using reference expressions and interpolation, we were able to use the output of the random\_pet resource as an input for the local\_file resource.

When Terraform creates these resources, it automatically understands the dependency. Since the local\_file resource depends on the output of the random\_pet resource, Terraform follows a specific order to provision them:

1. **Terraform first creates the random\_pet resource.**
2. **Then, it creates the local\_file resource.**

Similarly, when resources are deleted, Terraform follows the reverse order:

1. **Terraform first deletes the local\_file resource.**
2. **Then, it deletes the random\_pet resource.**

This type of dependency is called an **implicit dependency**. Here, we are not explicitly specifying which resource depends on another; Terraform automatically determines this based on the reference expression.

**Explicit Dependency**

There is another way to specify dependencies within the configuration file. Consider the case where we use the older configuration file without a reference expression for the file content. If we still want to ensure that the local\_file resource is created only after the random\_pet resource, we can achieve this using the depends\_on argument, as shown below:

resource "local\_file" "my\_file" {

filename = "/root/my\_pet.txt"

content = "My favorite pet is stored here."

depends\_on = [random\_pet.my\_pet]

}

Here, we have added the depends\_on argument inside the resource block for local\_file. We have provided a list of dependencies, which includes the random\_pet resource called my\_pet. This ensures that Terraform creates the local\_file resource **only after** the random\_pet resource is successfully created.

This type of dependency is called an **explicit dependency**.

**When to Use Explicit Dependencies**

Explicitly specifying a dependency is only necessary when a resource indirectly relies on another resource **but does not use a reference expression**. In such cases, Terraform does not automatically detect the dependency, so we must define it manually.

In the later sections of this course, we will explore real-world scenarios where explicit dependencies become essential.

**Conclusion**

Understanding the difference between **implicit** and **explicit** dependencies is crucial when working with Terraform. Implicit dependencies allow Terraform to determine the correct order automatically, while explicit dependencies ensure that resources are created in the correct sequence when indirect dependencies exist.