13 October

1) Create folder 13 Oct

2) Create main.tf file and copy storage account code from terraform registry

3) Dynamic block donot supports count, it actually only supports for_each

4) Diffrenece between set and list

In Terraform, `set` and `list` are both types of collections used to store multiple values, but they have some key differences:

1. Order:

- **List**: A list maintains the order of its elements. The order in which you add items to a list is preserved.
    - Example:
    - ```
      variable "my_list" {
        type = list(string)
        default = ["apple", "banana", "cherry"]
      }
      ```

      In this example, the order is: `apple`, `banana`, `cherry`.

- **Set**: A set **does not maintain order**. The values in a set are unordered, so they might appear in any order when you output them.
    - Example:
    - ```
      variable "my_set" {
        type = set(string)
        default = ["apple", "banana", "cherry"]
      }
      ```

      In this example, the order could be different each time, like `banana`, `apple`, `cherry`.

2. Duplicates:

- **List**: A list can have **duplicate** values. You can add the same value multiple times in a list.
    - Example:
    - ```
      variable "my_list" {
        type = list(string)
        default = ["apple", "apple", "banana"]
      }
      ```

      This list has two `apple` values.

- **Set**: A set **does not allow duplicates**. If you try to add the same value more than once, it will only appear once in the set.
    - Example:
    - ```
      variable "my_set" {
        type = set(string)
        default = ["apple", "apple", "banana"]
      ```

- `}`

This set will only store one `apple` value, so it will be: `apple`, `banana`.

3. Use cases:

- **List**: Use a list when the order of elements matters or when you want to allow duplicates.
- **Set**: Use a set when you want to ensure that the collection contains unique values and the order doesn't matter.

Summary:

- **List**: Ordered, allows duplicates.
- **Set**: Unordered, does not allow duplicates.