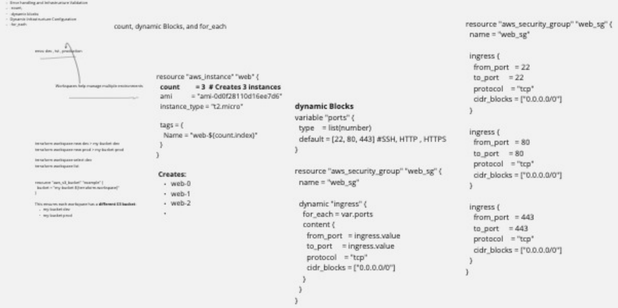

for_each in tf
iterate over maps or sets
if you need to dyanmically create IAM users or s3 buckets

```
variable "users" {
  type = map(string)
  default = {
    alice = "Admin"
    bob   = "Developer"
    charlie = "Tester"
  }
}

resource "aws_iam_user" "users" {
  for_each = var.users
  name     = each.key
  tags = {
    Role = each.value
  }
}
```

```
terraform_project/
├── main.tf
├── variables.tf
├── outputs.tf
├── terraform.tfvars
├── modules/
│   └── ec2_s3_module/
│       ├── main.tf
│       ├── variables.tf
│       ├── outputs.tf
│       ├── security_group.tf
│       └── iam.tf
```

Scenario:
You are a **DevOps Engineer** at a growing startup. Your team wants to deploy an **infrastructure-as-code (IaC) solution** for setting up the following AWS resources **across multiple environments (dev, staging, prod)** using **Terraform Workspaces, Modules, Variables, and Outputs.**

🎯 **Requirements:**
1️⃣ **Use Terraform Workspaces** to differentiate between **dev, staging, and prod** environments.
2️⃣ **Create a reusable Terraform module** for EC2 instances and S3 buckets.
3️⃣ The **EC2 instance** should:
- Use Amazon Linux 2
- Have an SSH security group
- Attach an IAM role to allow S3 access
- 4️⃣ The **S3 bucket** should:
- Have versioning enabled
- Be environment-specific (my-app-dev, my-app-staging, etc.)
- 5️⃣ **Use input variables** for instance type, region, and S3 bucket name.
- 6️⃣ **Use Terraform outputs** to display EC2 public IP and S3 bucket name.

Task:
✅ **Step 1:** Create a Terraform module (ec2_s3_module) that:
- Creates an **EC2 instance** with an IAM role allowing S3 access.
- Creates an **S3 bucket** with versioning enabled.

✅ **Step 2:** Use **Terraform Workspaces** to deploy this module for **dev, staging, and prod** environments.
✅ **Step 3:** Use **input variables** for EC2 instance type, region, and S3 bucket name.
✅ **Step 4:** Use **Terraform Outputs** to show:
- EC2 **public IP**
- S3 **bucket name**