Variables with priorities

Resource file:-

```
resource "aws_instance" "apache" {
  ami                    = var.aws_ami
  instance_type          = var.aws_instance_type
  tags = {
    Name = "apache"
  }
  root_block_device
    { volume_size =

  }
  user_data        = <<-EOF
  #!/bin/bash
  sudo -i
  yum install httpd -y
  echo "hello guys" >> /var/www/html/index.html
  systemctl start httpd
  systemctl enable httpd
  EOF
  count = var.count_number
}
```

var.tf file:-

```
variable "aws_instance_type"
  {type   = string
  default = "t2.micro"
}
variable "aws_ami"
  {type   = string
  default = "ami-07a6e3b1c102cdba8"
}
variable "volume_size"
  {type   = number
  default = 10
}
variable "count_number"
    {type = number
    default = 1
}
```

test.tfvars file:-

```
aws_instance_type = "t2.small"
volume_size = 8
count_number = 2
```

For .tfvars file we need to initiatialize terraform

terraform init

```
pratik@TheRock:/mnt/c/Users/pratik/Desktop/vaibhav$ tf init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.39.1...
- Installed hashicorp/aws v5.39.1 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
pratik@TheRock:/mnt/c/Users/pratik/Desktop/vaibhav$
```

terraform apply -var-file=test.tfvars

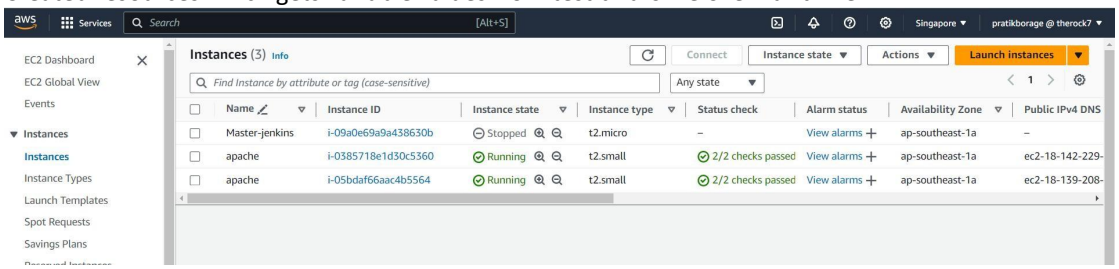
```
pratik@TheRock:/mnt/c/Users/pratik/Desktop/vaibhav$ tf apply -var-file=test.tfvars

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.apache[0] will be created
  + resource "aws_instance" "apache" {
      + ami                                  = "ami-07a6e3b1c102cdba8"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
```

Created resources which gets variable values from test.tfvars file over var.tf file



.auto.tfvars file:-

```
aws_instance_type = "t2.micro"
volume_size = 12
count_number = 3
```

For .auto.tfvars file - no need to initialize
Directly apply

It prioritizes .auto.tfvars file over test.tfvars and var.tf files

Created resources which gets variable values from .auto.tfvars file:-