Lesson 11 Demo 02

Working with Variables in Terraform

Objective: To utilize Terraform variables and local values for efficient and flexible infrastructure configurations

Tools required: Terraform, AWS, and Visual Studio Code

Prerequisites: Refer to the **Demo 01** of **Lesson 11** for creating access and secret key

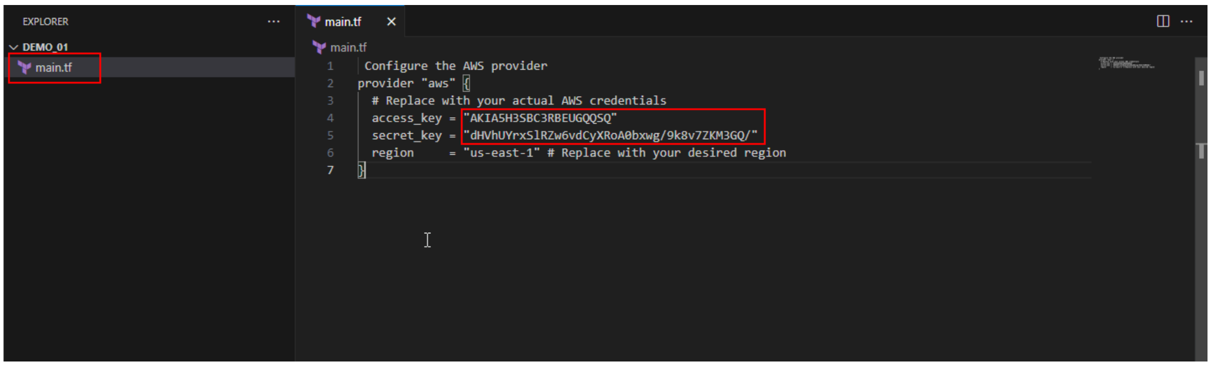
Steps to be followed:

1. Define local values and variables
2. Apply configuration using defined variables
3. Verify and utilize output values

Step 1: Define local values and variables

1.1 Open the Terraform configuration environment, create a file named **main.tf**, and add the following configuration block as shown in the screenshot below:

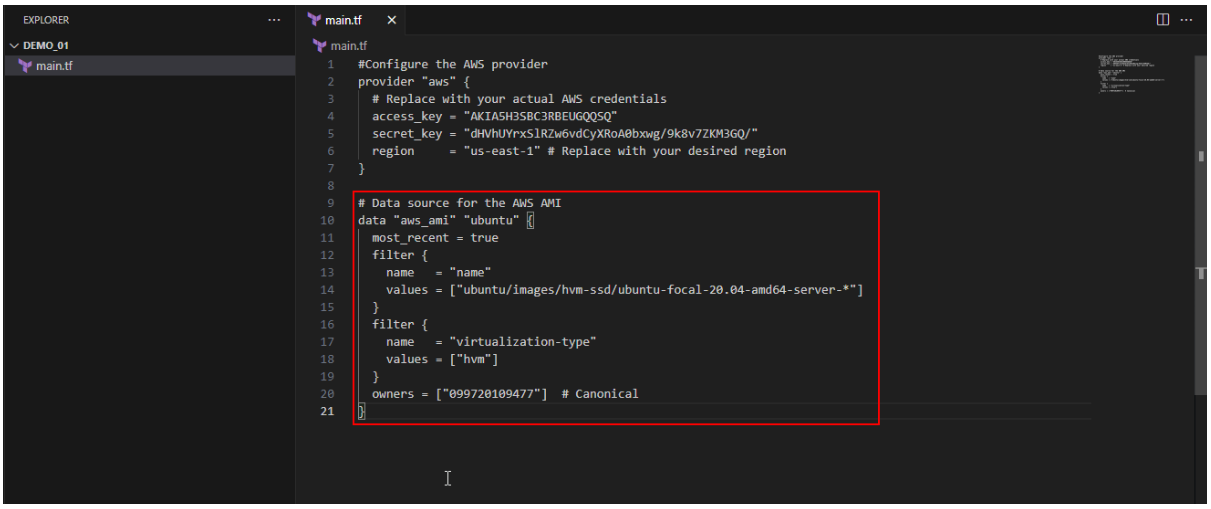
```
#Configure the AWS provider
provider "aws" {
  # Replace with your actual AWS credentials
  access_key = "YOUR_ACCESS_KEY"
  secret_key = "YOUR_SECRET_KEY"
  region     = "us-east-1" # Replace with your desired region
}
```



```
 Configure the AWS provider
provider "aws" {
  # Replace with your actual AWS credentials
  access_key = "AKIA5H3SBC3RBEUGQQSQ"
  secret_key = "dHVhUYrxSlRZw6vdCyXRoA0bxwg/9k8v7ZKM3GQ/"
  region     = "us-east-1" # Replace with your desired region
}
```

1.2 Add the following block to declare the AWS AMI data source as shown in the screenshot below:

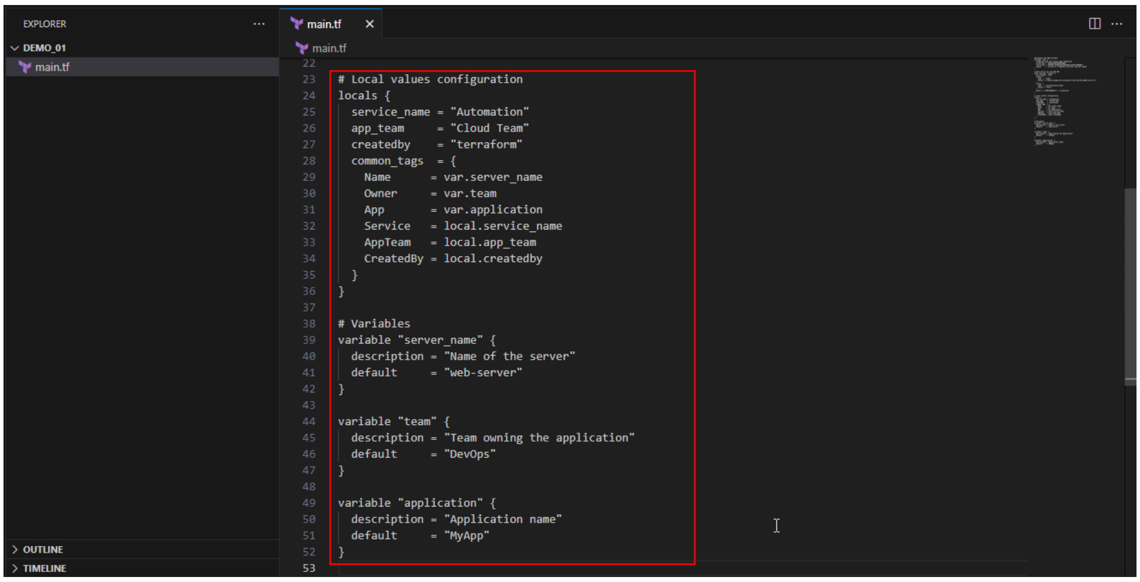
```
# Data source for the AWS AMI
data "aws_ami" "ubuntu" {
  most_recent = true
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
  owners = ["099720109477"]  # Canonical
}
```



```
#Configure the AWS provider
provider "aws" {
  # Replace with your actual AWS credentials
  access_key = "AKIA5H3SBC3RBEUGQQSQ"
  secret_key = "dHVhUYrxSlRZw6vdCyXRoA0bxwg/9k8v7ZKM3GQ/"
  region     = "us-east-1" # Replace with your desired region
}

# Data source for the AWS AMI
data "aws_ami" "ubuntu" {
  most_recent = true
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
  owners = ["099720109477"]  # Canonical
}
```

1.3 Insert the following block to define local values and variables that will be reused throughout your configuration:

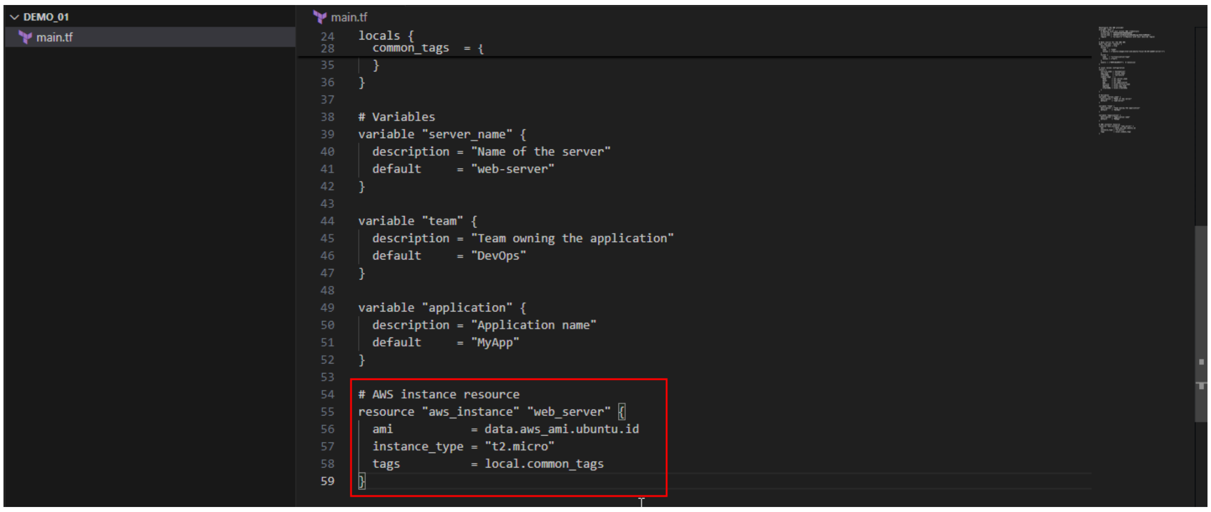
```
# Local values configuration
locals {
  service_name = "Automation"
  app_team     = "Cloud Team"
  createdby    = "terraform"
  common_tags  = {
    Name      = var.server_name
    Owner     = var.team
    App       = var.application
    Service   = local.service_name
    AppTeam   = local.app_team
    CreatedBy = local.createdby
  }
}

# Variables
variable "server_name" {
  description = "Name of the server"
  default     = "web-server"
}

variable "team" {
  description = "Team owning the application"
  default     = "DevOps"
}

variable "application" {
  description = "Application name"
  default     = "MyApp"
}
```



```
# Local values configuration
locals {
  service_name = "Automation"
  app_team     = "Cloud Team"
  createdby    = "terraform"
  common_tags  = {
    Name      = var.server_name
    Owner     = var.team
    App       = var.application
    Service   = local.service_name
    AppTeam   = local.app_team
    CreatedBy = local.createdby
  }
}

# Variables
variable "server_name" {
  description = "Name of the server"
  default     = "web-server"
}

variable "team" {
  description = "Team owning the application"
  default     = "DevOps"
}

variable "application" {
  description = "Application name"
  default     = "MyApp"
}
```

1.4 Add an AWS instance resource that utilizes the defined locals for tagging as shown in the screenshot below:

```
# AWS instance resource
resource "aws_instance" "web_server" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"
  tags          = local.common_tags
}
```



DEMO_01
main.tf

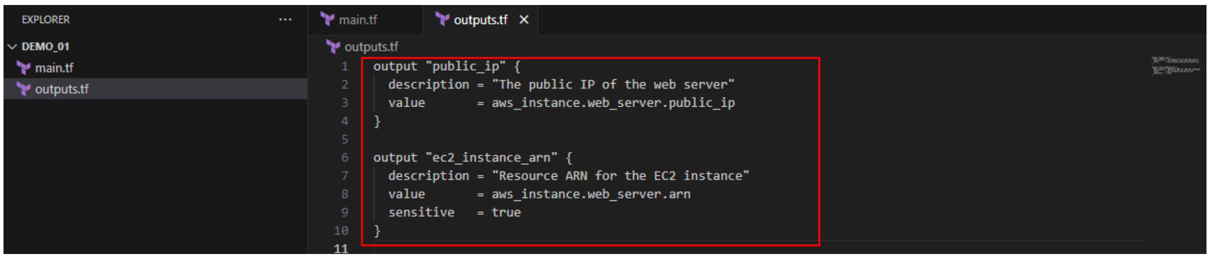
main.tf

```
locals {
  common_tags  = {
  }
}

# Variables
variable "server_name" {
  description = "Name of the server"
  default     = "web-server"
}

variable "team" {
  description = "Team owning the application"
  default     = "DevOps"
}

variable "application" {
  description = "Application name"
  default     = "MyApp"
}

# AWS instance resource
resource "aws_instance" "web_server" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"
  tags          = local.common_tags
}
```

Step 2: Apply configuration using defined variables

2.1 Create a file named **outputs.tf**, and add the following block to define outputs that will display important information post-deployment:

```
output "public_ip" {
  description = "The public IP of the web server"
  value       = aws_instance.web_server.public_ip
}

output "ec2_instance_arn" {
  description = "Resource ARN for the EC2 instance"
  value       = aws_instance.web_server.arn
  sensitive   = true
}
```



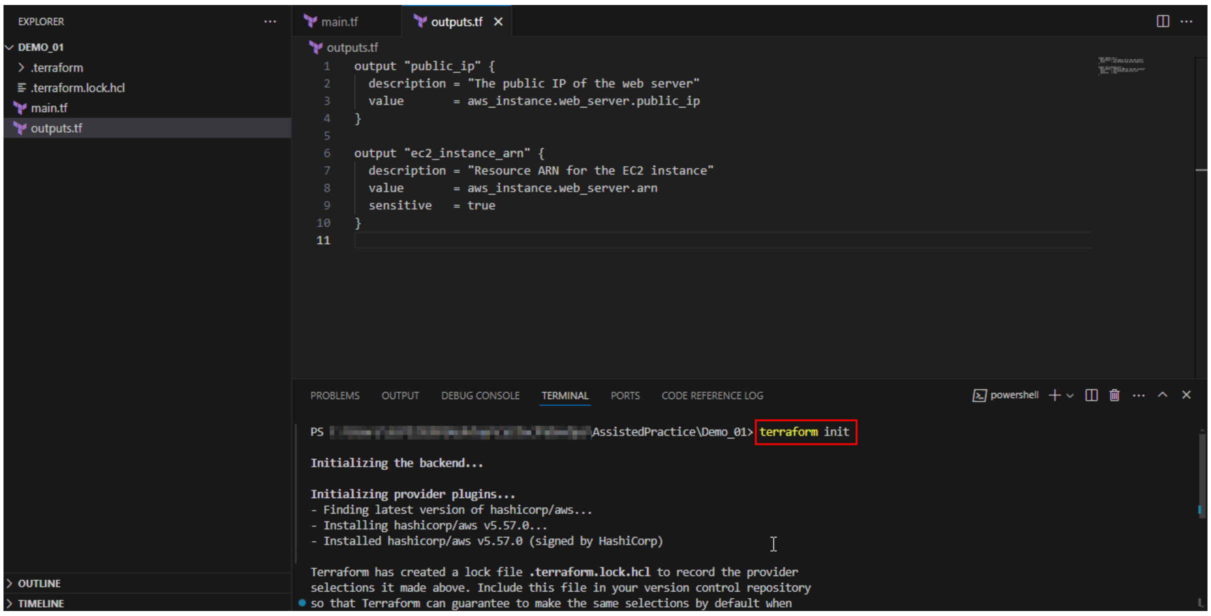
EXPLORER
DEMO_01
main.tf
outputs.tf

main.tf | outputs.tf

outputs.tf

```
output "public_ip" {
  description = "The public IP of the web server"
  value       = aws_instance.web_server.public_ip
}

output "ec2_instance_arn" {
  description = "Resource ARN for the EC2 instance"
  value       = aws_instance.web_server.arn
  sensitive   = true
}
```

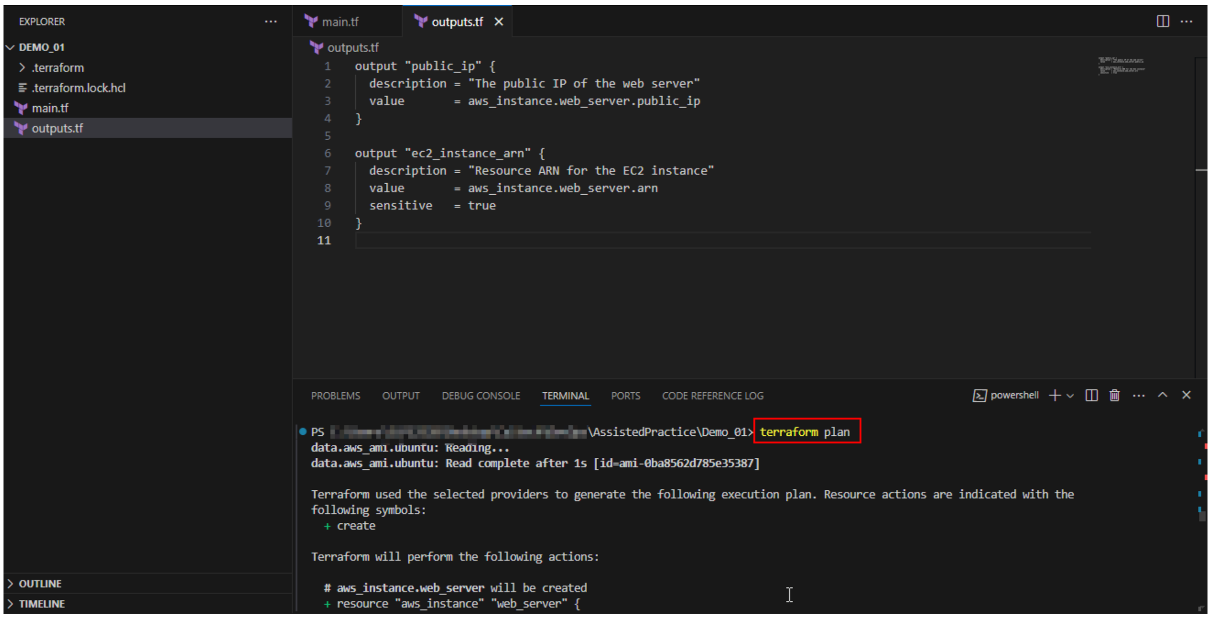
2.2 Initialize the Terraform configuration using the following command:
terraform init



```
output "public_ip" {
  description = "The public IP of the web server"
  value       = aws_instance.web_server.public_ip
}

output "ec2_instance_arn" {
  description = "Resource ARN for the EC2 instance"
  value       = aws_instance.web_server.arn
  sensitive   = true
}
```

```
PS [redacted]AssistedPractice\Demo_01> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.57.0...
- Installed hashicorp/aws v5.57.0 (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
```

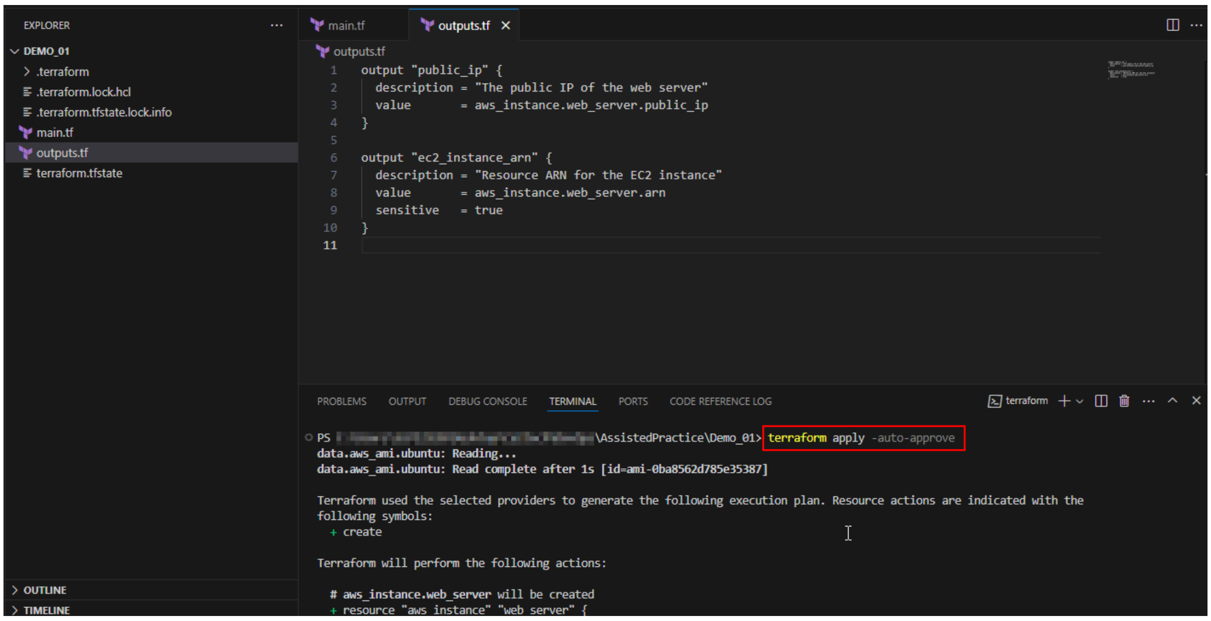
2.3 Plan the deployment using the following command to see the proposed changes:
terraform plan



```
output "public_ip" {
  description = "The public IP of the web server"
  value       = aws_instance.web_server.public_ip
}

output "ec2_instance_arn" {
  description = "Resource ARN for the EC2 instance"
  value       = aws_instance.web_server.arn
  sensitive   = true
}
```

```
PS [redacted]AssistedPractice\Demo_01> terraform plan
data.aws_ami.ubuntu: Reading...
data.aws_ami.ubuntu: Read complete after 1s [id=ami-0ba8562d785e35387]

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.web_server will be created
  + resource "aws_instance" "web_server" {
```

2.4 Apply the configuration using the following command to deploy the changes as shown in the screenshot below:
terraform apply -auto-approve

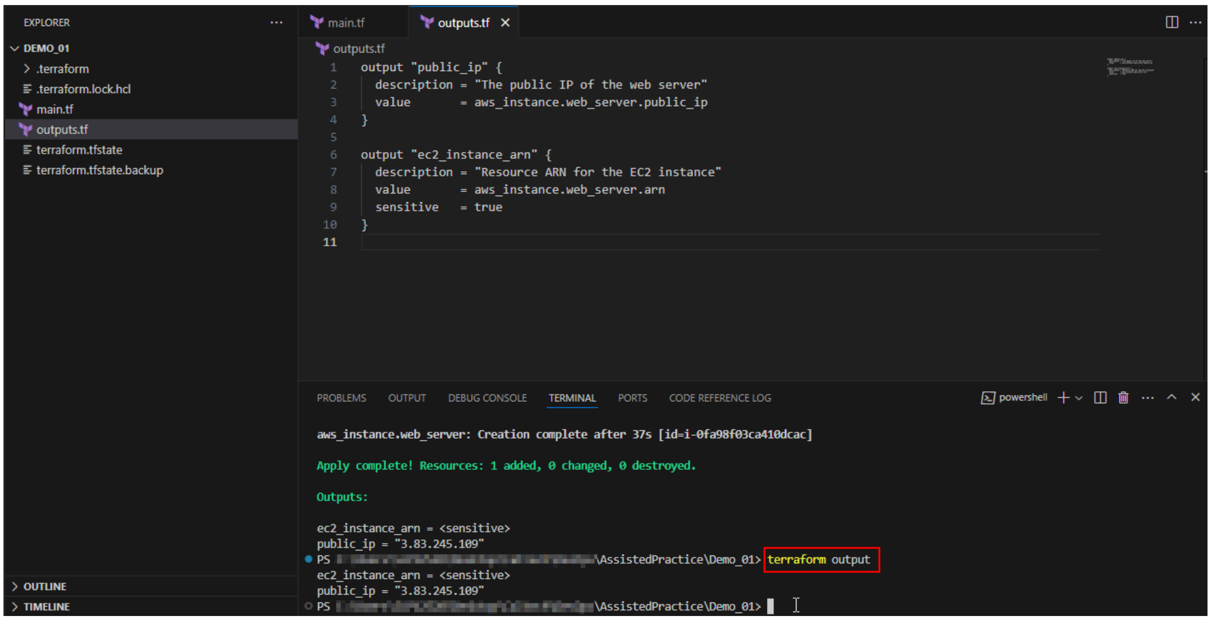


```
output "public_ip" {
  description = "The public IP of the web server"
  value       = aws_instance.web_server.public_ip
}

output "ec2_instance_arn" {
  description = "Resource ARN for the EC2 instance"
  value       = aws_instance.web_server.arn
  sensitive   = true
}
```

```
PS [redacted]\AssistedPractice\Demo_01> terraform apply -auto-approve
data.aws_ami.ubuntu: Reading...
data.aws_ami.ubuntu: Read complete after 1s [id=ami-0ba8562d785e35387]

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.web_server will be created
  + resource "aws_instance" "web_server" {
```

Step 3: Verify and utilize output values

3.1 Check the output values using the following command as shown in the screenshot below:
terraform output



```
output "public_ip" {
  description = "The public IP of the web server"
  value       = aws_instance.web_server.public_ip
}

output "ec2_instance_arn" {
  description = "Resource ARN for the EC2 instance"
  value       = aws_instance.web_server.arn
  sensitive   = true
}
```

```
aws_instance.web_server: Creation complete after 37s [id=i-0fa98f03ca410dcac]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

ec2_instance_arn = <sensitive>
public_ip = "3.83.245.109"
PS [redacted]\AssistedPractice\Demo_01> terraform output
ec2_instance_arn = <sensitive>
public_ip = "3.83.245.109"
PS [redacted]\AssistedPractice\Demo_01>
```

By following these steps, you have successfully utilized Terraform variables and local values for efficient and flexible infrastructure configurations.