



University of Petroleum and Energy Studies

System Provisioning and Configuration
Management Lab File

Name: Aanchal Tailwal

Faculty: Mrs. Silky Goel

Branch: BTech CSE

Batch: DevOps B-4 (NH)

Sap Id: 500097386

Roll No: R2142211334

INDEX

S.NO.	Name of Experiment	Page No.
1	Create an IAM account and download and install the Terraform.	3
2	Download Aws provider through terraform	4
3	Write the terraform script to perform some tasks	8
4	Write the terraform script to create the VPC.	9
5	Different types of Variables in Terraform	11
6	Write the script to create multiple instances using the variable having different configurations.	12
7	Write the steps for setting up the Ansible.	14
8	Write an ansible playbook to print hello-world using YML	15
9	Write an ansible playbook and ansible shell commands using yaml.	16
10	Write an ansible playbook to declare variables using yaml.	17
11	Write the playbook to print the default ipv4 address of each host along with the hosts name	18
12	Write the playbook to print the user defined variable from the command line.	19
13	Write the playbook to show the working of loops.	20
14	Write the playbook to pass the multiple variables from the command line.	22

EXPERIMENT-1

AIM : Terraform Installation

Step 1: go to <https://www.terraform.io/downloads.html>

Step 2: Scroll down a bit and you can see Windows. There will be 32-bit and 64-bit. As per your processor, click on the link and the zip file will start getting downloaded. It will be downloaded in your Downloads folder.

Step 3: Unzip the zip file.

Step 4: Open a command prompt and write the below commands: mkdir terraform cd terraform/

Step 5: The zipped file which has been unzipped in Step 3, copy that file and paste it in the folder directory created in step 4.

Step 6: In the command prompt window where you entered the commands mentioned in Step 4 just write the below command set PATH=%PATH%;C:\Users\Asus\terraform.

Step 7: Now write in the command prompt — terraform. Below is the screenshot of what it looks like in the machine.

Terraform 0.13 and later:

```

terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "example" {
  cidr_block = "10.0.0.0/16"
}

```

Search IAM

Dashboard

Access management

- User groups
- Users**
- Roles
- Policies
- Identity providers
- Account settings

Access reports

- Access Analyzer
- External access
- Unused access

Users (1/3) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Search

	User name	Path	Group	Last activity	MFA	Password age	Console last sign-in
<input type="checkbox"/>	terraform	/	0	-	-	-	-
<input checked="" type="checkbox"/>	terraform-user	/	0	20 minutes ago	-	1 hour	March 17, 2024, 00
<input type="checkbox"/>	test_user-1	/	1	15 days ago	-	15 days	March 01, 2024, 23

EXPERIMENT-2

Aim:

- Download aws provider through terraform.
- Provide the aws api access to terraform.
- Create an EC2 instance.
- Create an EC2 instance and add a tag.

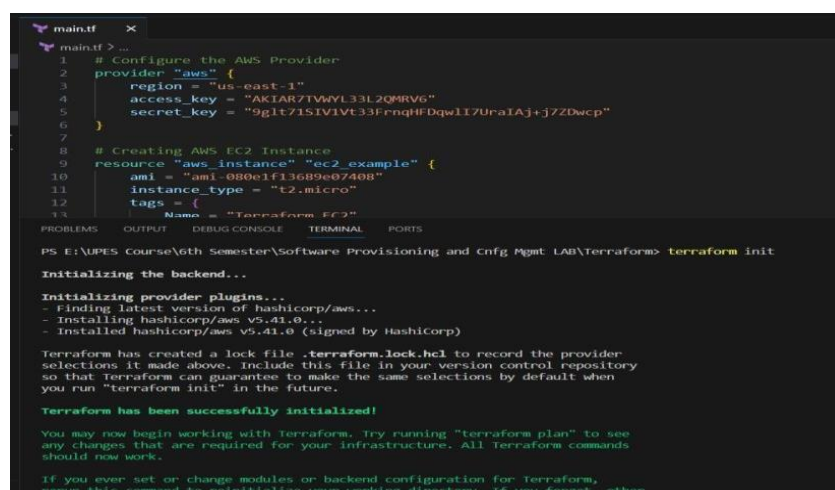
Created IAM Admin User — terraform-user

Then Created AWS API Access key for IAM User — terraform-user

```
resource "aws_instance" "spcm-lab1" {
  ami = "ami-0d3f444bc76de0a79"
  instance_type = "t2.micro"
  tags = {"Name":"terraformlab1"}
}
```

Terraform commands

- 1) **terraform init:** This command is responsible for downloading all the dependencies which are required for the provider AWS on your local machine.



```
main.tf
1 # Configure the AWS Provider
2 provider "aws" {
3   region = "us-east-1"
4   access_key = "AKIAR71VWYL33L2QMRV6"
5   secret_key = "9glt715IV1Vt33FrnqHFDqwlI7UraIAj+j7ZDwcp"
6 }
7
8 # Creating AWS EC2 Instance
9 resource "aws_instance" "ec2_example" {
10  ami = "ami-080e1f13689e07408"
11  instance_type = "t2.micro"
12  tags = {
13    Name = "Terraform EC2"
14  }
15 }

PS E:\UPES Course\6th Semester\Software Provisioning and Cnfg Mgmt LAB\Terraform> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.41.0...
- Installed hashicorp/aws v5.41.0 (signed by HashiCorp)

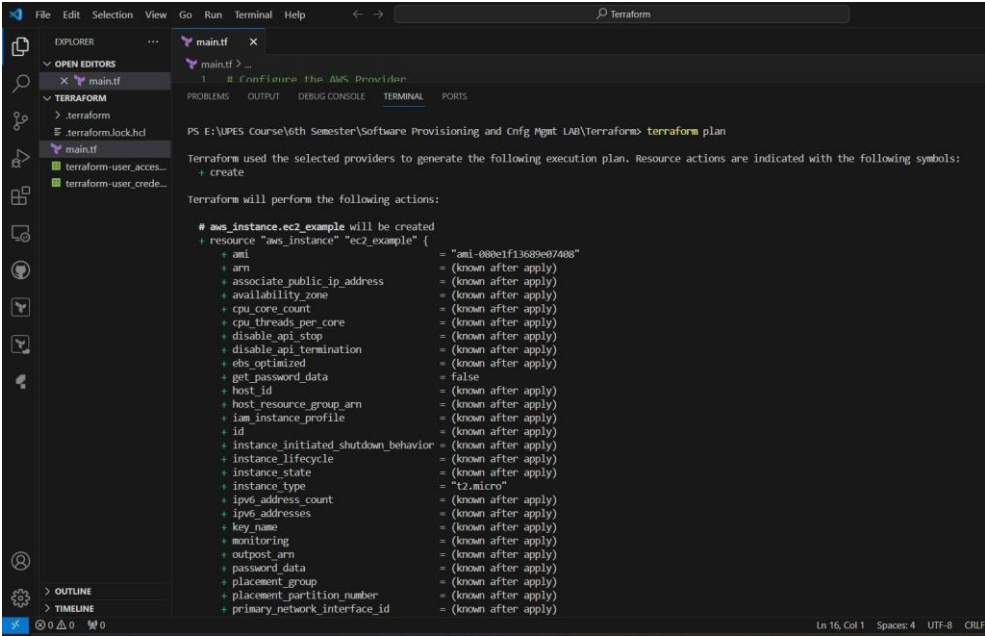
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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
```

- 2) **terraform plan:** This command will help you to understand how many resources you are gonna add or delete.



```

PS E:\VUPES Course\6th Semester\Software Provisioning and Cnfg Mgmt LAB\Terraform> terraform plan

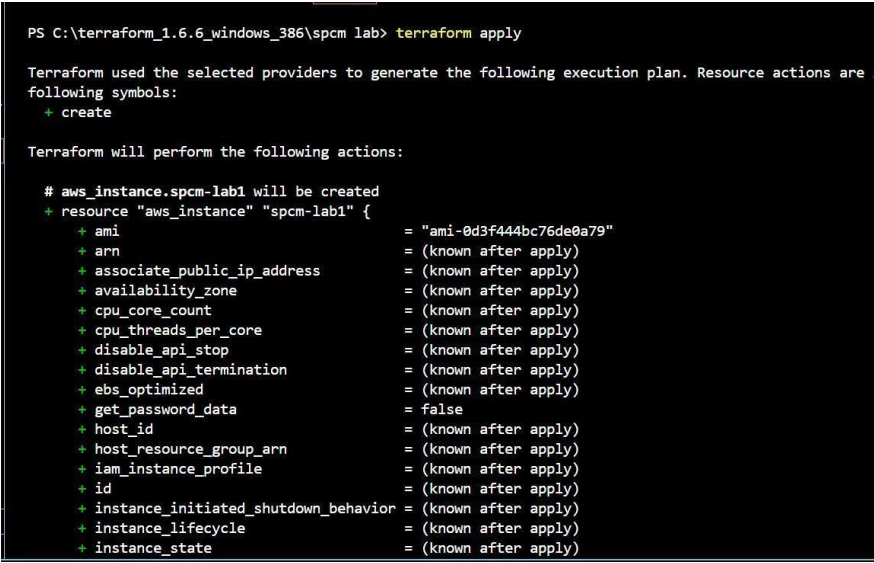
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.ec2_example will be created
+ resource "aws_instance" "ec2_example" {
  + ami               = "ami-088e1f13689e07408"
  + 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)
  + disable_api_termination = (known after apply)
  + ebs_optimized      = (known after apply)
  + get_password_data  = false
  + host_id            = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile = (known after apply)
  + id                 = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle = (known after apply)
  + instance_state     = (known after apply)
  + instance_type      = "t2.micro"
  + ipv6_address_count = (known after apply)
  + ipv6_addresses     = (known after apply)
  + key_name            = (known after apply)
  + monitoring          = (known after apply)
  + outpost_arn         = (known after apply)
  + password_data       = (known after apply)
  + placement_group     = (known after apply)
  + placement_partition_number = (known after apply)
  + primary_network_interface_id = (known after apply)

```

- 3) **terraform apply:** This command will do some real stuff on AWS. Once you issue this command, it will be going to connect to AWS and then finally going to provision AWS instances.



```

PS C:\terraform_1.6.6_windows_386\spcm lab> terraform apply

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

Terraform will perform the following actions:

# aws_instance.spcm-lab1 will be created
+ resource "aws_instance" "spcm-lab1" {
  + ami               = "ami-0d3f444bc76de0a79"
  + 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)
  + disable_api_termination = (known after apply)
  + ebs_optimized      = (known after apply)
  + get_password_data  = false
  + host_id            = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile = (known after apply)
  + id                 = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle = (known after apply)
  + instance_state     = (known after apply)

```

```

}
+ tags_all
+ "Name" = "terraformlab1"
}
+ tenancy
+ user_data
+ user_data_base64
+ user_data_replace_on_change
+ vpc_security_group_ids
}

Plan: 1 to add, 0 to change, 0 to destroy.

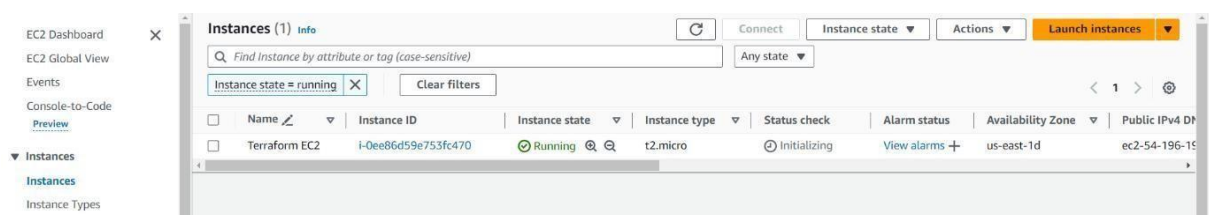
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.spcm-lab1: Creating...
aws_instance.spcm-lab1: Still creating... [10s elapsed]
aws_instance.spcm-lab1: Still creating... [20s elapsed]
aws_instance.spcm-lab1: Still creating... [30s elapsed]
aws_instance.spcm-lab1: Creation complete after 34s [id=i-0240f85e17f4c4a43]

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

```



4) terraform destroy: It will remove all the running EC2 Instances.

```

main.tf > terraform destroy
PS E:\UPES Course\6th Semester\Software Provisioning and Cnfg Mgmt LAB\Terraform> terraform destroy
aws_instance.ec2_example: Refreshing state... [id=i-0ee86d59e753fc470]

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

Terraform will perform the following actions:

# aws_instance.ec2_example will be destroyed
- resource "aws_instance" "ec2_example" {
  ami           = "ami-0800cf13609007408" -> null
  arn           = "arn:aws:ec2:us-east-1:136612870135:instance/i-0ee86d59e753fc470" -> null
  associate_public_ip_address = true -> null
  availability_zone = "us-east-1d" -> null
  cpu_core_count = 1 -> null
  cpu_threads_per_core = 1 -> null
  disable_api_stop = false -> null
  disable_api_termination = false -> null
  ebs_optimized = false -> null
  get_password_data = false -> null
  hibernation = false -> null
  id = "i-0ee86d59e753fc470" -> null
  instance_initiated_shutdown_behavior = "stop" -> null
  instance_state = "running" -> null
  instance_type = "t2.micro" -> null
  ip_address_count = 0 -> null
  ip_addresses = [] -> null
  monitoring = false -> null
  placement_partition_number = 0 -> null
  primary_network_interface_id = "eni-05283b2395eb31ff1" -> null
  private_dns = "ip-172-31-29-129.ec2.internal" -> null
  private_ip = "172.31.29.129" -> null
  public_dns = "ec2-54-196-196-69.compute-1.amazonaws.com" -> null
  public_ip = "54.196.196.69" -> null
  secondary_private_ips = [] -> null
  security_groups = []
}

```

```

main.tf > terraform destroy
PS E:\UPES Course\6th Semester\Software Provisioning and Cnfg Mgmt LAB\Terraform> terraform destroy
aws_instance.ec2_example: Refreshing state... [id=i-0ee86d59e753fc470]

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

Terraform will perform the following actions:

# aws_instance.ec2_example will be destroyed
- resource "aws_instance" "ec2_example" {
  ami           = "ami-0800cf13609007408" -> null
  arn           = "arn:aws:ec2:us-east-1:136612870135:instance/i-0ee86d59e753fc470" -> null
  associate_public_ip_address = true -> null
  availability_zone = "us-east-1d" -> null
  cpu_core_count = 1 -> null
  cpu_threads_per_core = 1 -> null
  disable_api_stop = false -> null
  disable_api_termination = false -> null
  ebs_optimized = false -> null
  get_password_data = false -> null
  hibernation = false -> null
  id = "i-0ee86d59e753fc470" -> null
  instance_initiated_shutdown_behavior = "stop" -> null
  instance_state = "running" -> null
  instance_type = "t2.micro" -> null
  ip_address_count = 0 -> null
  ip_addresses = [] -> null
  monitoring = false -> null
  placement_partition_number = 0 -> null
  primary_network_interface_id = "eni-05283b2395eb31ff1" -> null
  private_dns = "ip-172-31-29-129.ec2.internal" -> null
  private_ip = "172.31.29.129" -> null
  public_dns = "ec2-54-196-196-69.compute-1.amazonaws.com" -> null
  public_ip = "54.196.196.69" -> null
  secondary_private_ips = [] -> null
  security_groups = []
}

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_instance.ec2_example: Destroying... [id=i-0ee86d59e753fc470]
aws_instance.ec2_example: Still destroying... [id=i-0ee86d59e753fc470, 10s elapsed]
aws_instance.ec2_example: Still destroying... [id=i-0ee86d59e753fc470, 20s elapsed]
aws_instance.ec2_example: Still destroying... [id=i-0ee86d59e753fc470, 30s elapsed]
aws_instance.ec2_example: Still destroying... [id=i-0ee86d59e753fc470, 40s elapsed]
aws_instance.ec2_example: Destruction complete after 45s

Destroy complete! Resources: 1 destroyed.
PS E:\UPES Course\6th Semester\Software Provisioning and Cnfg Mgmt LAB\Terraform>

```

EC2 Dashboard

EC2 Global View

Events

Console-to-Code

Preview

▼ Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Instances (1) Info

Refresh

Connect

Instance state ▼

Actions ▼

Launch instances ▼

Find Instance by attribute or tag (case-sensitive)

Any state ▼

< 1 > ⌕

<input type="checkbox"/>	Name ↗	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status	Availability Zone ▼	Public IP
<input type="checkbox"/>	Terraform EC2	i-0ee86d59e753fc470	Terminated ⌕	t2.micro	–	View alarms +	us-east-1d	–

Select an instance

Experiment- 3

Aim:

Write the terraform script to perform the following task:

- create a S3 bucket
- enable the version of the bucket
- add an upload folder

Terraform script:

```
provider "aws" {  
  access_key = "AKIAVURU2DW5CV6PQ7EV"  
  secret_key = "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"  
  region = "ap-south-1"  
}  
  
resource "aws_s3_bucket" "lab_5_sanskar" {  
  bucket = "lab-5-sanskar"  
  
  tags = {  
    "Name": "Lab5-sanskar"  
  }  
}
```

```
resource "aws_s3_bucket_versioning" "lab_5_versioning" {  
  bucket = aws_s3_bucket.lab_5_sanskar.id  
  
  versioning_configuration {  
    status = "Enabled"  
  }  
}
```


EXPERIMENT- 4

AIM: Create Security Group on AWS using Terraform

The image displays three sequential screenshots of a terminal window running Terraform commands to create an AWS security group.

First Screenshot: Shows the initial state of the Terraform directory. The file explorer on the left lists files: `terraform`, `terraform.lock.hcl`, `aws.provider.tf`, `terraform.tfstate`, and `terraform.tfstate.backup`. The terminal shows the command `terraform apply` being executed. The output indicates that Terraform will create the `aws_security_group.web` resource.

```

saurin@DESKTOP-H80V70: ~/Desktop/Terraform (master)
$ terraform apply

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_security_group.web will be created
+ resource "aws_security_group" "web" {
+   description = "Allow HTTP and SSH inbound traffic"
+   ingress {
+     cidr_blocks = [
+       "0.0.0.0/0",
+     ]
+     description = ""
+     from_port = 0
+     to_port = 0
+     protocol = "tcp"
+     security_groups = []
+     self = false
+     to_port = 0
+   },
+   id = (known after apply)
+   ingress {
+     cidr_blocks = [
+       "0.0.0.0/0",
+     ]
+     description = ""
+     from_port = 22
+     to_port = 22
+     protocol = "tcp"
+     security_groups = []
+     self = false
+     to_port = 0
+   },
+   tags = {}
+ }

```

Second Screenshot: Shows the Terraform configuration file `aws.provider.tf` being edited. The file defines the AWS provider and the security group resource.

```

1 provider "aws" {
2   region = "eu-north-1" # Change this to your desired AWS region
3 }
4
5 resource "aws_security_group" "web" {
6   name = "web-sg"
7   description = "Allow HTTP and SSH inbound traffic"
8
9   // Inbound rule for HTTP (port 80)
10  ingress {
11    from_port = 80
12    to_port = 80
13    protocol = "tcp"
14    cidr_blocks = ["0.0.0.0/0"]
15  }
16
17  // Inbound rule for SSH (port 22)
18  ingress {
19    from_port = 22
20    to_port = 22
21    protocol = "tcp"
22    cidr_blocks = ["0.0.0.0/0"]
23  }
24
25  // Outbound rule allowing all traffic to leave
26  egress {
27    from_port = 0
28    to_port = 0
29    protocol = "all"
30    cidr_blocks = ["0.0.0.0/0"]
31  }
32
33  tags = {
34    Name = "web-sg"
35  }
36 }
37

```

Third Screenshot: Shows the final output of the `terraform apply` command. The security group `aws_security_group.web` has been successfully created. The output includes the resource ID and the tags assigned to it.

```

Plan: 1 to add, 0 to change, 0 to destroy.
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
Enter a value: yes

aws_security_group.web: Creating...
aws_security_group.web: Creation complete after 4s [cloudwatchlogs:aws:logs]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
$ terraform apply

```

The screenshot displays the AWS Management Console interface for a security group named 'web_sg' with ID 'sg-02b4479f03dbf6e3e'. The console is in the 'eu-north-1' region. The left sidebar shows navigation options like EC2 Dashboard, Instances, Images, and Elastic Block Store. The main content area is titled 'sg-02b4479f03dbf6e3e - web_sg' and includes an 'Actions' dropdown. The 'Details' section shows the security group's name, ID, description ('Allow HTTP and SSH inbound traffic'), VPC ID ('vpc-0af5297bf9e39888c'), owner ('211125740082'), and rule counts (2 inbound, 1 outbound). Below this, the 'Inbound rules' tab is active, showing a table with 2 rules. The table has columns for Name, Security group rule..., IP version, Type, Protocol, and Port range. The first rule is for HTTP (Type: HTTP, Protocol: TCP, Port range: 80) and the second is for SSH (Type: SSH, Protocol: TCP, Port range: 22). The bottom of the screen shows a Windows taskbar with various application icons and a system clock indicating 17:05 on 17-03-2024.

eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#SecurityGroup:sg-02b4479f03dbf6e3e

sg-02b4479f03dbf6e3e - web_sg

Details

Security group name web_sg	Security group ID sg-02b4479f03dbf6e3e	Description Allow HTTP and SSH inbound traffic	VPC ID vpc-0af5297bf9e39888c
Owner 211125740082	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules (2)

	Name	Security group rule...	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-0e065c6f52efea65d	IPv4	HTTP	TCP	80
<input type="checkbox"/>	-	sgr-0c151595047bcaaff	IPv4	SSH	TCP	22

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

17:05 17-03-2024

EXPERIMENT- 5

Aim:

Write the terraform script to create the VPC.

```
resource "aws_vpc" "sanskar_public_vpc" {
  cidr_block = "10.0.0.0/24"
  instance_tenancy = "default"
  tags = {
    Name = "sanskar-public-vpc"
  }
}
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

Terraform will perform the following actions:

```
# aws_vpc.sanskar_public_vpc will be created
+ resource "aws_vpc" "sanskar_public_vpc" {
  + arn                               = (known after apply)
  + cidr_block                       = "10.0.0.0/24"
  + default_network_acl_id           = (known after apply)
  + default_route_table_id           = (known after apply)
  + default_security_group_id        = (known after apply)
  + dhcp_options_id                  = (known after apply)
  + enable_dns_hostnames              = (known after apply)
  + enable_dns_support                = true
  + enable_network_address_usage_metrics = (known after apply)
  + id                               = (known after apply)
  + instance_tenancy                 = "default"
  + ipv6_association_id              = (known after apply)
  + ipv6_cidr_block                   = (known after apply)
  + ipv6_cidr_block_network_border_group = (known after apply)
  + main_route_table_id              = (known after apply)
  + owner_id                         = (known after apply)
  + tags                             = {
    + "Name" = "sanskar-public-vpc"
  }
  + tags_all                          = {
    + "Name" = "sanskar-public-vpc"
  }
}
```

```
+ tags = {
  + "Name" = "sanskar-public-vpc"
}
+ tags_all = {
  + "Name" = "sanskar-public-vpc"
}
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_vpc.sanskar_public_vpc: Creating...

aws_vpc.sanskar_public_vpc: Creation complete after 2s [id=vpc-03b672a9081f166e2]

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

Your VPCs (1/1)

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP
sanskar-public-vpc	vpc-036c759432ca68353	Available	10.0.0.0/24	-	dhcp-0e4f9ceb8870db590

vpc-036c759432ca68353 / sanskar-public-vpc

Details	Resource map	CIDRs	Flow logs	Tags	Integrations
VPC ID vpc-036c759432ca68353 Tenancy Default Default VPC No Network Address Usage metrics Disabled	State Available DHCP option set dhcp-0e4f9ceb8870db590 IPv4 CIDR 10.0.0.0/24 Route 53 Resolver DNS Firewall rule groups	DNS hostnames Disabled Main route table rtb-038b04c1a7f387f93 IPv6 pool - Owner ID 387731561914	DNS resolution Enabled Main network ACL acl-0952b1ab6cca4af981 IPv6 CIDR (Network border group) -		

EXPERIMENT- 6 : Write the script to create multiple instances using the variable having different configurations.

Aim:

1) Variables

The screenshot shows a Visual Studio Code editor with a Terraform script in `main.tf` and its execution output in the terminal.

main.tf Script:

```

1 variable "template" {
2   type = string
3   default = "01000000-0000-4000-8000-000030080200"
4 }
5 output "string_output" {
6   value = var.template
7 }
8
9 variable "users" {
10  type = list
11  default = ["root", "user1", "user2"]
12 }
13
14 output "list_output" {
15   value = var.users
16 }
17
18 variable "plans" {
19   type = map
20   default = {
21     "5USD" = "1xCPU-1GB"
22     "10USD" = "1xCPU-2GB"
23     "20USD" = "2xCPU-4GB"
24   }
25 }
26 output "Maps_output" {
27   value = var.plans
28 }
29
30 variable "set_password" {
31   default = false
32 }
33 output "Boolean_output" {
34   value = var.set_password
35 }

```

Terminal Output:

```

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

Outputs:
Boolean_output = false
Maps_output = tomap({
  "10USD" = "1xCPU-2GB"
  "20USD" = "2xCPU-4GB"
  "5USD" = "1xCPU-1GB"
})
list_output = tolist([
  "root",
  "user1",
  "user2",
])
string_output = "01000000-0000-4000-8000-000030080200"

```

Create multiple instances using the variable having different configurations

The screenshot displays a Terraform IDE interface with two panels. The top panel shows the Terraform configuration file (`main.tf`) for `Exp-4.3`. The bottom panel shows the console view of the Terraform state, displaying a list of instances created by the configuration.

Terraform Configuration (main.tf):

```

1 provider "aws" {
2   region = "us-east-1"
3 }
4
5 variable "configs" {
6   type = list(object({
7     application_name = string
8     ami              = string
9     no_of_instances = number
10    instance_type    = string
11  })
12  default = [
13    {
14      "application_name": "App-1",
15      "ami" : "ami-080e1f13689e07408",
16      "no_of_instances" : "2",
17      "instance_type" : "t2.micro",
18    },
19    {
20      "application_name": "2-dev",
21      "ami" : "ami-080e1f13689e07408",
22      "instance_type" : "t2.micro",
23      "no_of_instances" : "1",
24    },
25    {
26      "application_name": "3-dev",
27      "ami" : "ami-080e1f13689e07408",
28      "instance_type" : "t2.micro",
29      "no_of_instances" : "3"
30    }
31  ]
32 }

```

Terraform Configuration (main.tf):

```

33 }
34
35 locals {
36   s_configs = [
37     for srv in var.configs : [
38       for i in range(1, srv.no_of_instances + 1) : {
39         instance_name = "${srv.application_name}-${i}"
40         instance_type = srv.instance_type
41         ami           = srv.ami
42       }
43     ]
44   }
45
46   locals {
47     instances = flatten(local.s_configs)
48   }
49
50 resource "aws_instance" "lab_4_part_2" {
51   for_each = { for inst in local.instances : inst.instance_name => inst }
52   ami      = each.value.ami
53   instance_type = each.value.instance_type
54   tags = {
55     Name = each.value.instance_name
56   }
57 }
58
59 output "instances" {
60   value = aws_instance.lab_4_part_2
61   description = "instances created in part-2"
62 }
63

```

Console View:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
App-1-1	i-00943b2d1a0aec613	Running	t2.micro	Initializing	View alarms +	us-east-1c
3-dev-3	i-0063991cac0b095bd	Running	t2.micro	Initializing	View alarms +	us-east-1c
3-dev-2	i-0292d427b3e588119	Running	t2.micro	Initializing	View alarms +	us-east-1c
3-dev-1	i-02f952e2ae8df9ccd	Running	t2.micro	Initializing	View alarms +	us-east-1c
App-1-2	i-03005c16916c04eb1	Running	t2.micro	Initializing	View alarms +	us-east-1c
2-dev-1	i-0a5ece51eca50be3b	Running	t2.micro	Initializing	View alarms +	us-east-1c

EXPERIMENT 7

Write the steps for setting up the Ansible.

- sudo apt-get -y upgrade && sudo apt-get -y update

```
root@Sanskar:~# sudo apt-get -y upgrade && sudo apt-get -y update
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  python3-update-manager ubuntu-advantage-tools update-manager-core
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Hit:1 http://ppa.launchpad.net/ansible/ansible/ubuntu focal InRelease
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:3 http://archive.ubuntu.com/ubuntu focal InRelease
Get:4 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [2821 kB]
Hit:6 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Get:7 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [952 kB]
Fetched 4000 kB in 4s (1084 kB/s)
Reading package lists... Done
root@Sanskar:~# sudo apt install software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
software-properties-common is already the newest version (0.99.9.12).
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
root@Sanskar:~# sudo apt-add-repository --yes --update ppa:ansible/ansible
Hit:1 http://ppa.launchpad.net/ansible/ansible/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:4 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
root@Sanskar:~# sudo apt install ansible
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ansible-core python3-bcrypt python3-jmespath python3-kerberos python3-ntlm-auth python3-packaging python3-paramiko
  python3-pyparsing python3-requests-kerberos python3-requests-ntlm python3-resolvlib python3-winrm python3-xlrd python3-xmldict
  sshpass
0 upgraded, 15 newly installed, 0 to remove and 3 not upgraded.
Need to get 32.3 MB of archives.
After this operation, 323 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-pyparsing all 2.4.6-1 [61.3 kB]
Get:2 http://ppa.launchpad.net/ansible/ansible/ubuntu focal/main amd64 python3-resolvlib all 0.5.4-2ppa-focal [12.6 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-packaging all 20.3-1 [26.8 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-jmespath all 0.9.4-2ubuntu1 [21.5 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-kerberos amd64 1.1.14-3.1build1 [22.6 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-ntlm-auth all 1.1.0-1 [19.6 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-bcrypt amd64 3.1.7-2ubuntu1 [30.4 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-paramiko all 2.6.0-2ubuntu0.3 [123 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-requests-kerberos all 0.12.0-2 [11.9 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-requests-ntlm all 1.1.0-1 [6000 B]
Get:11 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-xlrd all 0.12.0-1 [12.6 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-winrm all 0.3.0-2 [21.7 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-sshpass amd64 1.06-1 [10.5 kB]
Get:14 http://ppa.launchpad.net/ansible/ansible/ubuntu focal/main amd64 ansible all 5.10.0-1ppa-focal [21.0 MB]
Get:15 http://ppa.launchpad.net/ansible/ansible/ubuntu focal/main amd64 ansible-core all 5.10.0-1ppa-focal [21.0 MB]
Fetched 32.3 MB in 5s (6704 kB/s)
Selecting previously unselected package python3-pyparsing.
(Reading database ... 32760 files and directories currently installed.)
Preparing to unpack .../00-python3-pyparsing_2.4.6-1_all.deb ...
Unpacking python3-pyparsing (2.4.6-1) ...
Selecting previously unselected package python3-packaging.
Preparing to unpack .../01-python3-packaging_20.3-1_all.deb ...
Unpacking python3-packaging (20.3-1) ...
Selecting previously unselected package python3-resolvlib.
Preparing to unpack .../02-python3-resolvlib_0.5.4-2ppa-focal_all.deb ...
Unpacking python3-resolvlib (0.5.4-2ppa-focal) ...
Selecting previously unselected package ansible-core.
Preparing to unpack .../03-ansible-core_5.10.0-1ppa-focal_all.deb ...
Unpacking ansible-core (5.10.0-1ppa-focal) ...
Selecting previously unselected package ansible.
Preparing to unpack .../04-ansible_5.10.0-1ppa-focal_all.deb ...
Unpacking ansible (5.10.0-1ppa-focal) ...
Selecting previously unselected package python3-jmespath.
Preparing to unpack .../05-python3-jmespath_0.9.4-2ubuntu1_all.deb ...
Unpacking python3-jmespath (0.9.4-2ubuntu1) ...
Selecting previously unselected package python3-requests-kerberos.
Preparing to unpack .../06-python3-requests-kerberos_1.1.14-3.1build1_amd64.deb ...
Unpacking python3-requests-kerberos (1.1.14-3.1build1) ...
```

- sudo apt install software-properties-common

```
0 upgraded, 15 newly installed, 0 to remove and 3 not upgraded.
Need to get 32.3 MB of archives.
After this operation, 323 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-pyparsing all 2.4.6-1 [61.3 kB]
Get:2 http://ppa.launchpad.net/ansible/ansible/ubuntu focal/main amd64 python3-resolvlib all 0.5.4-2ppa-focal [12.6 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-packaging all 20.3-1 [26.8 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-jmespath all 0.9.4-2ubuntu1 [21.5 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-kerberos amd64 1.1.14-3.1build1 [22.6 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-ntlm-auth all 1.1.0-1 [19.6 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-bcrypt amd64 3.1.7-2ubuntu1 [30.4 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-paramiko all 2.6.0-2ubuntu0.3 [123 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-requests-kerberos all 0.12.0-2 [11.9 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-requests-ntlm all 1.1.0-1 [6000 B]
Get:11 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-xlrd all 0.12.0-1 [12.6 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-winrm all 0.3.0-2 [21.7 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-sshpass amd64 1.06-1 [10.5 kB]
Get:14 http://ppa.launchpad.net/ansible/ansible/ubuntu focal/main amd64 ansible all 5.10.0-1ppa-focal [21.0 MB]
Get:15 http://ppa.launchpad.net/ansible/ansible/ubuntu focal/main amd64 ansible-core all 5.10.0-1ppa-focal [21.0 MB]
Fetched 32.3 MB in 5s (6704 kB/s)
Selecting previously unselected package python3-pyparsing.
(Reading database ... 32760 files and directories currently installed.)
Preparing to unpack .../00-python3-pyparsing_2.4.6-1_all.deb ...
Unpacking python3-pyparsing (2.4.6-1) ...
Selecting previously unselected package python3-packaging.
Preparing to unpack .../01-python3-packaging_20.3-1_all.deb ...
Unpacking python3-packaging (20.3-1) ...
Selecting previously unselected package python3-resolvlib.
Preparing to unpack .../02-python3-resolvlib_0.5.4-2ppa-focal_all.deb ...
Unpacking python3-resolvlib (0.5.4-2ppa-focal) ...
Selecting previously unselected package ansible-core.
Preparing to unpack .../03-ansible-core_5.10.0-1ppa-focal_all.deb ...
Unpacking ansible-core (5.10.0-1ppa-focal) ...
Selecting previously unselected package ansible.
Preparing to unpack .../04-ansible_5.10.0-1ppa-focal_all.deb ...
Unpacking ansible (5.10.0-1ppa-focal) ...
Selecting previously unselected package python3-jmespath.
Preparing to unpack .../05-python3-jmespath_0.9.4-2ubuntu1_all.deb ...
Unpacking python3-jmespath (0.9.4-2ubuntu1) ...
Selecting previously unselected package python3-requests-kerberos.
Preparing to unpack .../06-python3-requests-kerberos_1.1.14-3.1build1_amd64.deb ...
Unpacking python3-requests-kerberos (1.1.14-3.1build1) ...
```

-sudo apt-add-repository --yes --update ppa:ansible/ansible

-sudo apt install ansible

```
Setting up python3-pyparsing (2.4.6-1) ...
Setting up python3-jmespath (0.9.4-2ubuntu1) ...
Setting up python3-requests-kerberos (0.12.0-2) ...
Setting up python3-paramiko (2.6.0-2ubuntu0.3) ...
Setting up python3-requests-ntlm (1.1.0-1) ...
Setting up python3-packaging (20.3-1) ...
Setting up python3-winrm (0.3.0-2) ...
Setting up ansible-core (2.12.10-1ppa-focal) ...
Setting up ansible (5.10.0-1ppa-focal) ...
Processing triggers for man-db (2.9.1-1) ...
root@Sanskar:~# ansible --version
ansible [core 2.12.10]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.8.10 (default, Nov 22 2023, 10:22:35) [GCC 9.4.0]
  jinja version = 2.10.1
  libyaml = True
root@Sanskar:~#
```


EXPERIMENT- 8

WRITE AN ANSIBLE PLAYBOOK TO PRINT HELLO-WORLD USING YML

```

GNU nano 4.8                                hello.yml
---
- name: Generic Playbook Name
  hosts: "{{ target_hosts | default('localhost') }}"
  tasks:
    - name: Print Hello, World!
      debug:
        msg: "Hello, World!"

root@Sanskar:~# nano hello.yml
root@Sanskar:~# ansible-playbook hello.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Generic Playbook Name] *****
TASK [Gathering Facts] *****
ok: [localhost]

TASK [Print Hello, World!] *****
ok: [localhost] => {
  "msg": "Hello, World!"
}

PLAY RECAP *****
localhost : ok=2  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

root@Sanskar:~#

```


EXPERIMENT- 9

WRITE AN ANSIBLE PLAYBOOK AND ANSIBLE SHELL COMMAND USING YAML.

```

GNU nano 4.8
---
- name: Execute Shell Command
  hosts: localhost
  tasks:
    - name: Run a Shell Command
      shell: echo "Hello, World!"
      register: shell_output

    - name: Display Output
      debug:
        var: shell_output.stdout

root@Sanskar:~# nano hello.yml
root@Sanskar:~# nano exp2.yml
root@Sanskar:~# ansible-playbook exp2.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Execute Shell Command] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Run a Shell Command] *****
changed: [localhost]

TASK [Display Output] *****
ok: [localhost] => {
  "shell_output.stdout": "Hello, World!"
}

PLAY RECAP *****
localhost                : ok=3  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

EXPERIMENT- 10

WRITE AN ANSIBLE PLAYBOOK TO DECLARE THE VARIABLE USING YAML

```

---
- name: Example playbook with variable declaration
  hosts: localhost
  vars:
    message: "Hello, World!"
    number: 42
    fruits:
      - apple
      - banana
      - orange
  tasks:
    - name: Print message variable
      debug:
        msg: "{{ message }}"

    - name: Print number variable
      debug:
        msg: "{{ number }}"

    - name: Print fruits variable
      debug:
        msg: "{{ fruits }}"

```

```

root@Sanskar:~# nano exp2.yml
root@Sanskar:~# nano exp3.yml
root@Sanskar:~# ansible-playbook exp3.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Example playbook with variable declaration] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Print message variable] *****
ok: [localhost] => {
  "msg": "Hello, World!"
}

TASK [Print number variable] *****
ok: [localhost] => {
  "msg": 42
}

TASK [Print fruits variable] *****
ok: [localhost] => {
  "msg": [
    "apple",
    "banana",
    "orange"
  ]
}

PLAY RECAP *****
localhost : ok=4  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

EXPERIMENT -11

1. Write a playbook to print the default ipv4 address of each host along with the host name.

```

root@Sanskar:~# nano /etc/ansible/hosts
root@Sanskar:~# ansible-playbook -i /etc/ansible/hosts print_ipv4.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Print Default IPv4 Addresses of Hosts] *****
skipping: no hosts matched

PLAY RECAP *****

```

Command -: \$ ansible localhost -u ansible -m setup

```

root@Sanskar:~# ansible localhost -u ansible -m setup
localhost | SUCCESS => {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": [
      "172.19.30.14"
    ],
    "ansible_all_ipv6_addresses": [
      "fe80::215:5dff:feda:65e4"
    ],
    "ansible_apparmor": {
      "status": "disabled"
    },
    "ansible_architecture": "x86_64",
    "ansible_bios_date": "",
    "ansible_bios_vendor": "",
    "ansible_bios_version": "",
    "ansible_board_asset_tag": "",
    "ansible_board_name": "",
    "ansible_board_serial": "",
    "ansible_board_vendor": "",
    "ansible_board_version": "",
    "ansible_chassis_asset_tag": "",
    "ansible_chassis_serial": "",
    "ansible_chassis_vendor": "",
    "ansible_chassis_version": "",
    "ansible_cmdline": {
      "WSL_ROOT_INIT": "1",

```

EXPERIMENT- 12

Write a playbook to print the user defined variables from the command line.

```

---
- name: Print User-Defined Variables
  hosts: localhost
  gather_facts: false
  vars:
    user_defined_var1: "{{ user_var1 }}"
    user_defined_var2: "{{ user_var2 }}"

  tasks:
    - name: Display User-Defined Variables
      debug:
        msg: "User-defined Variable 1: {{ user_defined_var1 }}, User-defined Variable 2: {{ user_defined_var2 }}"

```

```

root@Sanskar:~# ansible-playbook -i localhost, print_user_vars.yml -e "user_var1=value1 user_var2=value2"

PLAY [Print User-Defined Variables] *****

TASK [Display User-Defined Variables] *****
ok: [localhost] => {
  "msg": "User-defined Variable 1: value1, User-defined Variable 2: value2"
}

PLAY RECAP *****
localhost : ok=1  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

```

---
- name: Display Gym and Yoga Products
  hosts: localhost
  gather_facts: false

  tasks:
    - name: Display Gym Product
      debug:
        msg: "Gym Product: {{ gym_product }}"

    - name: Display Yoga Product
      debug:
        msg: "Yoga Product: {{ yoga_product }}"

```

```

root@Sanskar:~# nano multivariableplaybook.yml
root@Sanskar:~# ansible-playbook -i localhost multivariableplaybook.yml -e "gym_product=Dumbbells" -e "yoga_product=Yoga Mat"

[WARNING]: Unable to parse /root/localhost as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Display Gym and Yoga Products] *****

TASK [Display Gym Product] *****
ok: [localhost] => {
  "msg": "Gym Product: Dumbbells"
}

TASK [Display Yoga Product] *****
ok: [localhost] => {
  "msg": "Yoga Product: Yoga"
}

PLAY RECAP *****
localhost : ok=2  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

EXPERIMENT- 13

Write the playbook to show the working of loops.

```
- name: Generate Gym Products List
  hosts: localhost
  gather_facts: false

  vars:
    gym_products:
      - name: "Dumbbells"
        weight: "5kg"
        price: "$20"
      - name: "Jump Rope"
        length: "Adjustable"
        price: "$10"
      - name: "Yoga Mat"
        size: "Standard"
        price: "$15"
      - name: "Resistance Bands"
        level: "Medium"
        price: "$25"

  tasks:
    - name: Display Gym Products
      debug:
        msg: |
          Product: {{ item.name }}
          {% if item.weight is defined %}
```

root@Sanskar:~# nano print_user_vars.yml
root@Sanskar:~# nano loopingplaybook.yml
root@Sanskar:~# ansible-playbook -i localhost loopingplaybook.yml
[WARNING]: Unable to parse /root/localhost as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

```
PLAY [Generate Gym Products List] *****

TASK [Display Gym Products] *****
ok: [localhost] => (item={'name': 'Dumbbells', 'weight': '5kg', 'price': '$20'}) => {
  "msg": "Product: Dumbbells\nWeight: 5kg\nPrice: $20\n"
}
ok: [localhost] => (item={'name': 'Jump Rope', 'length': 'Adjustable', 'price': '$10'}) => {
  "msg": "Product: Jump Rope\nLength: Adjustable\nPrice: $10\n"
}
ok: [localhost] => (item={'name': 'Yoga Mat', 'size': 'Standard', 'price': '$15'}) => {
  "msg": "Product: Yoga Mat\nSize: Standard\nPrice: $15\n"
}
ok: [localhost] => (item={'name': 'Resistance Bands', 'level': 'Medium', 'price': '$25'}) => {
  "msg": "Product: Resistance Bands\nLevel: Medium\nPrice: $25\n"
}

PLAY RECAP *****
localhost          : ok=1   changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

EXPERIMENT- 14

Write a playbook to print the user defined variables from the command line.

```

---
- name: Print User-Defined Variables
  hosts: localhost
  gather_facts: false
  vars:
    user_defined_var1: "{{ user_var1 }}"
    user_defined_var2: "{{ user_var2 }}"

  tasks:
    - name: Display User-Defined Variables
      debug:
        msg: "User-defined Variable 1: {{ user_defined_var1 }}, User-defined Variable 2: {{ user_defined_var2 }}"

```

```

root@Sanskar:~# ansible-playbook -i localhost, print_user_vars.yml -e "user_var1=value1 user_var2=value2"

PLAY [Print User-Defined Variables] *****

TASK [Display User-Defined Variables] *****
ok: [localhost] => {
  "msg": "User-defined Variable 1: value1, User-defined Variable 2: value2"
}

PLAY RECAP *****
localhost : ok=1  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

```

---
- name: Display Gym and Yoga Products
  hosts: localhost
  gather_facts: false

  tasks:
    - name: Display Gym Product
      debug:
        msg: "Gym Product: {{ gym_product }}"

    - name: Display Yoga Product
      debug:
        msg: "Yoga Product: {{ yoga_product }}"

```

```

root@Sanskar:~# nano multivariableplaybook.yml
root@Sanskar:~# ansible-playbook -i localhost multivariableplaybook.yml -e "gym_product=Dumbbells" -e "yoga_product=Yoga Mat"

[WARNING]: Unable to parse /root/localhost as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Display Gym and Yoga Products] *****

TASK [Display Gym Product] *****
ok: [localhost] => {
  "msg": "Gym Product: Dumbbells"
}

TASK [Display Yoga Product] *****
ok: [localhost] => {
  "msg": "Yoga Product: Yoga"
}

PLAY RECAP *****
localhost : ok=2  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```