

Assignment 1

Create a file called main.tf.

Create a docker image resource and call it nginx_image.

Set the name of the image to nginx:latest.

Save and exit the file.

Initialize Terraform.

Plan the deploy and output a terraform plan called tf_image_plan.

Apply the plan using tf_image_plan.

Code:

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.12.0"
    }
  }
}

provider "docker" {
  host = "npipe:////./pipe/docker_engine"
}

resource "docker_image" "nginx_image" {
  name = "nginx:latest"
}
```

Command:

terraform init

terraform plan -out tf_image_plan

terraform apply "tf_image_plan"

Assignment-2

Create a new Terraform file called main.tf.

Create three variables.

The first variable, called image_name, needs to be set to ghost:latest.

The second variable is called container_name with a default of ghost_blog.

The final variable is called ext_port and set the default to port 80.

Create a Docker image resource called ghost_image that uses the image_name variable.

Create a Docker container resource called ghost_container.

The name will use the container_name variable.

The image will use the ghost_image resource.

The internal port will be set to 2368.

The external port will use ext_port variable.

Initialize Terraform.

Create a Terraform plan that uses the following variables:

container_name = ghost_blog1

image_name = ghost:alpine

ext_port = 8080

Output the plan to a file called tfplan.

Then apply the plan using tfplan and make sure that the apply doesn't prompt for input.

```
variable "image_name" {
  default = "ghost:latest"
}

variable "container_name" {
  default = "ghost_blog"
}

variable "ext_port" {
  default = "80"
}

terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.12.0"
    }
  }
}

provider "docker" {
  host = "npipe:///.//pipe//docker_engine"
}
```

```

resource "docker_container" "ghost_container" {
  name = var.container_name
  image = docker_image.ghost_image.latest

  ports {
    internal = "2368"
    external = var.ext_port
  }
}

resource "docker_image" "ghost_image" {
  name = var.image_name
}

```

Commands:

terraform init

terraform plan -var 'container_name=ghost_blog1' -out tfplan

terraform plan -var 'image_name=ghost:alpine' -out tfplan

terraform plan -var 'ext_port=8080' -out tfplan

terraform apply "tfplan"

Assignment-3

Adding Maps and Lookups in your Terraform files

The lab files can be found @ <https://github.com/satyensingh/terraform-assignment-resources.git>

In the repo you will find main.tf, outputs.tf, and variables.tf.

Add a new variable called env. Set a description to "env: dev or prod".

Convert the type from image_name to map.

Change the default to use key/value pairs. Set dev to ghost:latest and prod to ghost:alpine.

Convert container_name to a map. Change the default to use key/value pairs. Set dev to blog_dev and prod to blog_prod.

Convert ext_port to a map. Change the default to use key/value pairs. Set dev to 8080 and prod to 80.

Now initialize Terraform.

Setup the Development environment

Create a workspace called dev.

Generate a Terraform plan. Output the plan and call it tfdev_plan. Pass in a variable called env and set it to dev.

Apply tfdev_plan.

Setup the Production environment

Create a workspace called prod.

Generate a Terraform plan. Output the plan and call it tfprod_plan. Pass in a variable called env and set it to prod.

Apply tfprod_plan.

Verify both environments work

Open a browser and navigate to the public IP. This should pull up the production environment.

Open a browser tab and navigate to the public IP on port 8080. This should pull up the development environment.

Main.tf

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.12.0"
    }
  }
}

provider "docker" {
  host = "npipe:////./pipe//docker_engine"
}

# Download the latest Ghost Image
resource "docker_image" "image_id" {
  name = "${lookup(var.image_name, var.env)}"
}

# Start the Container
resource "docker_container" "container_id" {
  name      = "${lookup(var.container_name, var.env)}"
  image     = "${docker_image.image_id.latest}"
  ports {
    internal = "2368"
    external = "${lookup(var.ext_port, var.env)}"
  }
}
```

```
}  
}
```

Variable.tf

```
variable "env" {  
  description = "env: dev or prod"  
  default = "dev"  
}  
  
variable "image_name" {  
  description = "Image for container."  
  type = map(string)  
  default = {  
    dev : "ghost:latest"  
    prod : "ghost:alpine"  
  }  
}  
  
variable "container_name" {  
  description = "Name of the container."  
  type = map(string)  
  default = {  
    dev : "blog_dev"  
    prod : "blog_prod"  
  }  
}  
  
variable "ext_port" {  
  description = "External port for container."  
  type = map(string)  
  default = {  
    dev : 8080  
    prod : 80  
  }  
}
```

Output.tf

```
#Output the IP Address of the Container
```

```

output "ip_address" {
  value          = "${docker_container.container_id.ip_address}"
  description = "The IP for the container."
}

output "container_name" {
  value          = "${docker_container.container_id.name}"
  description = "The name of the container."
}

```

Commands:

```

terraform init
terraform workspace new dev
terraform plan -var 'env=dev' -out tfdev_plan
terraform apply "tfdev_plan"

```

```

terraform workspace new prod
terraform plan -var 'env=prod' -out tfprod_plan
terraform apply "tfprod_plan"

```

```

terraform workspace list
docker ps

```

Assignment-4

Create Ghost module

Create a directory called ghost.

Your modules will be made up of three files: main.tf, variables.tf and outputs.tf.

main.tf

In main.tf you will deploy out two resources `docker_image` and `docker_container`. The `docker_image` resource name will be `ghost_image`.

The name will use the `image_name` variable.

The `docker_container` resource name will be `ghost_container`.

The name will be set using a variable called `container_name`. The image will be set using `docker_image.ghost_image.latest`. Set the external port to use the `ext_port` variable.

```
main.tf X
ghost > main.tf
1 terraform {
2   required_providers {
3     docker = {
4       source = "kreuzwerker/docker"
5       version = "2.12.0"
6     }
7   }
8 }
9
10 provider "docker" {
11   host = "npipe:////./pipe/docker_engine"
12 }
13
14 resource "docker_container" "ghost_container" {
15   name = var.container_name
16   image = docker_image.ghost_image.latest
17
18   ports {
19     internal = "2368"
20     external = var.ext_port
21   }
22 }
23
24 resource "docker_image" "ghost_image" {
25   name = var.image_name
26 }
```

variables.tf

In variables.tf create three variables: image_name, container_name and ext_port.

```
main.tf  variable.tf X
ghost > variable.tf
1  variable "image_name" {
2    //default = "ghost:latest"
3  }
4
5  variable "container_name" {
6    //default = "ghost_blog"
7  }
8
9  variable "ext_port" {
10   //default = "80"
11 }
```

outputs.tf

In outputs.tf create two outputs: ip and container_name. The ip output the ghost_container's ip_address attribute.

The container_name output the ghost_container's name attribute.

```
main.tf  outputs.tf X
ghost > outputs.tf
1  #Output the IP Address of the Container
2  output "ip_address" {
3    value      = docker_container.ghost_container.ip_address
4    description = "The IP for the container."
5  }
6
7  output "container_name" {
8    value      = docker_container.ghost_container.name
9    description = "The name of the container."
10 }
```

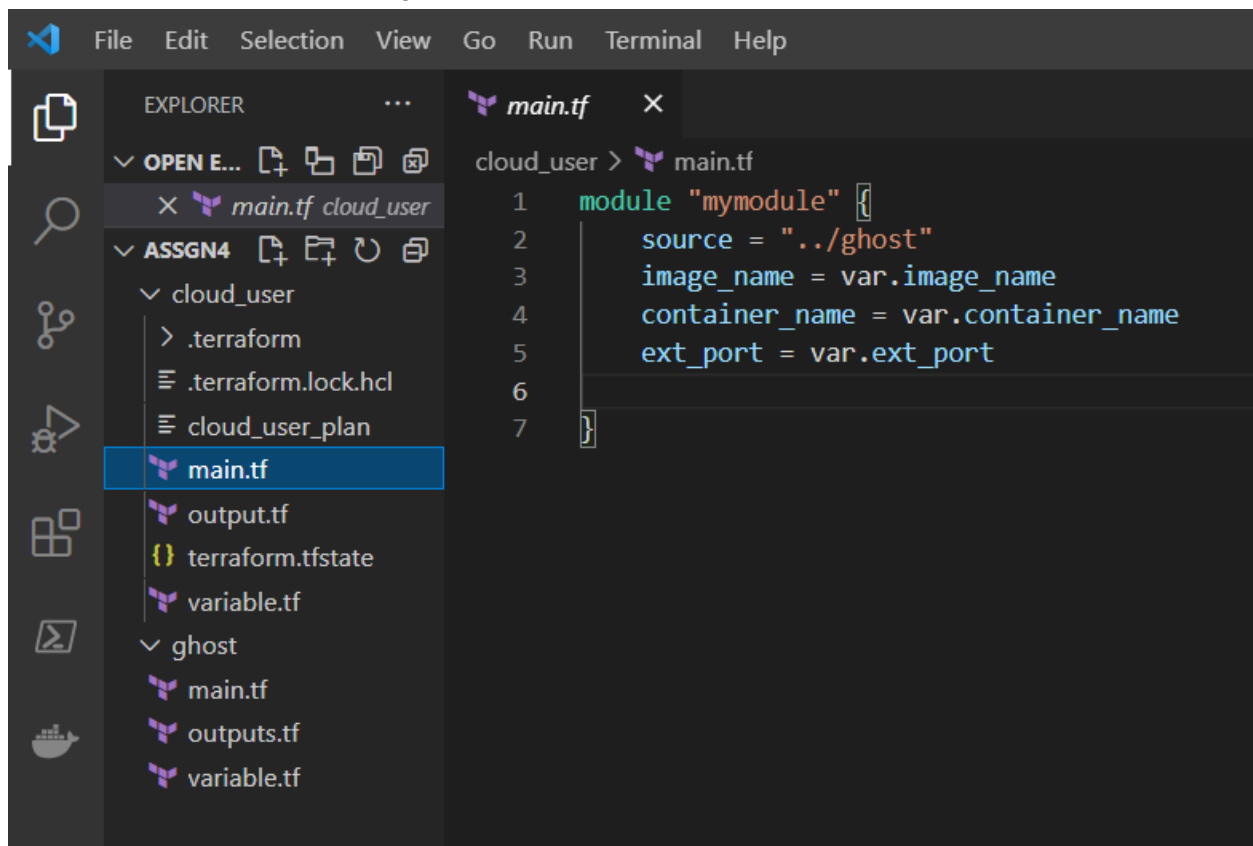
Create root module

main.tf

In [cloud_user](#) directory create main.tf, variables.tf and outputs.tf. In main.tf will use the ghost module.

Set image_name using a variable called image_name.

Set container_name using a variable called container_name.
Set ext_port using a variable called ext_port.
In variables.tf create three image_name, container_name and ext_port.

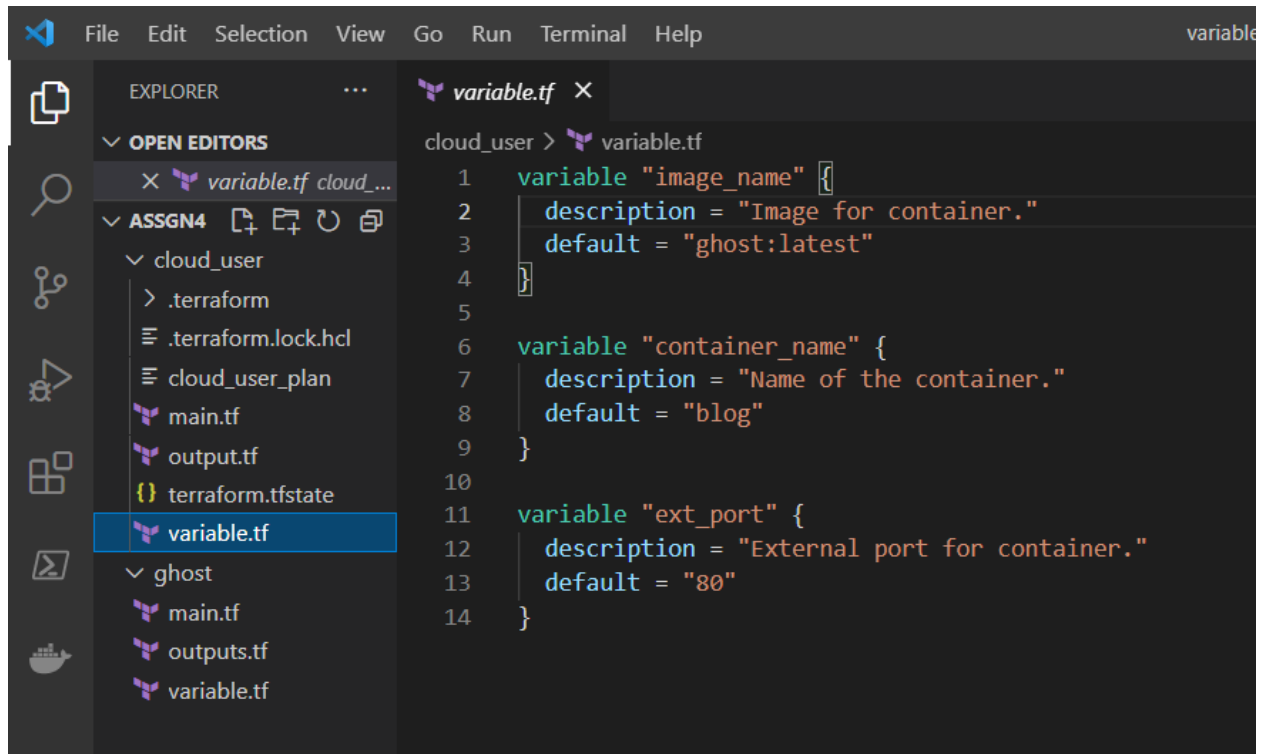


variables.tf

The image_name will have a default value of ghost:latest with a description of Image for container.

The container_name will have a default value of blog with a description of Name of the container.

The ext_port will have a default value of 80 with a description of External port for container.

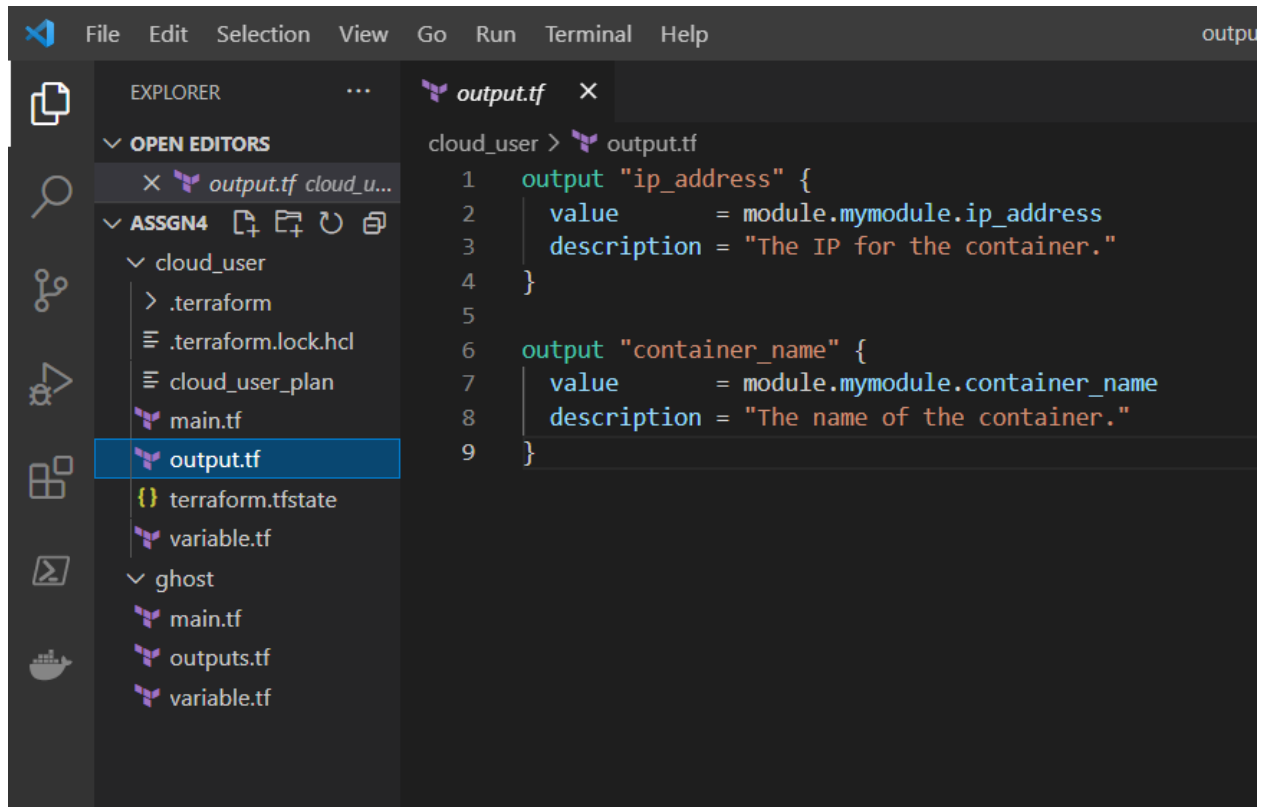
A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with folders 'cloud_user' and 'ghost'. Under 'cloud_user', there are files '.terraform', '.terraform.lock.hcl', 'cloud_user_plan', 'main.tf', 'output.tf', and 'terraform.tfstate'. The 'variable.tf' file is selected and highlighted. The main editor window displays the content of 'variable.tf', which defines three variables: 'image_name' (default 'ghost:latest'), 'container_name' (default 'blog'), and 'ext_port' (default '80').

```
1  variable "image_name" {
2      description = "Image for container."
3      default = "ghost:latest"
4  }
5
6  variable "container_name" {
7      description = "Name of the container."
8      default = "blog"
9  }
10
11 variable "ext_port" {
12     description = "External port for container."
13     default = "80"
14 }
```

outputs.tf

In outputs.tf create two outputs: ip and container_name. The ip output the ghost_container's ip_address attribute.

The container_name output the ghost_container's name attribute.



Deploy the infrastructure

Initialize Terraform.

Generate a Terraform plan and output a plan file.

Deploy the infrastructure using the plan file.

Commands:

```
cd cloud_user
```

```
terraform init
```

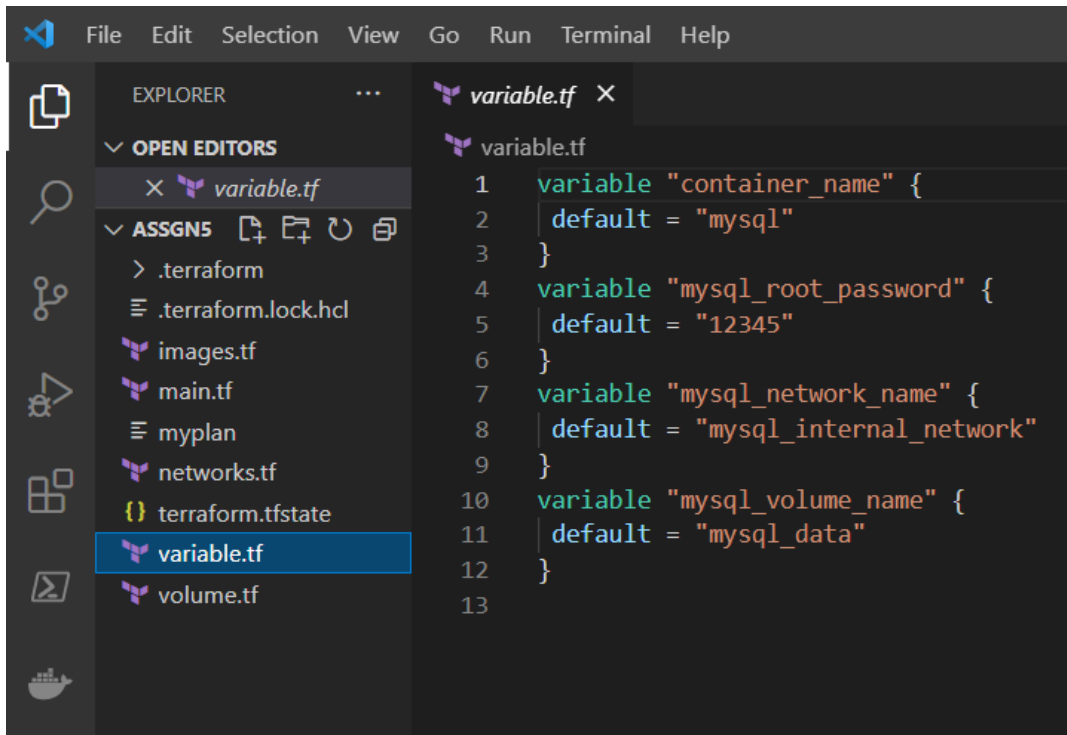
```
terraform plan -out cloud_user_plan
```

```
terraform apply "cloud_user_plan"
```

Assignment-5

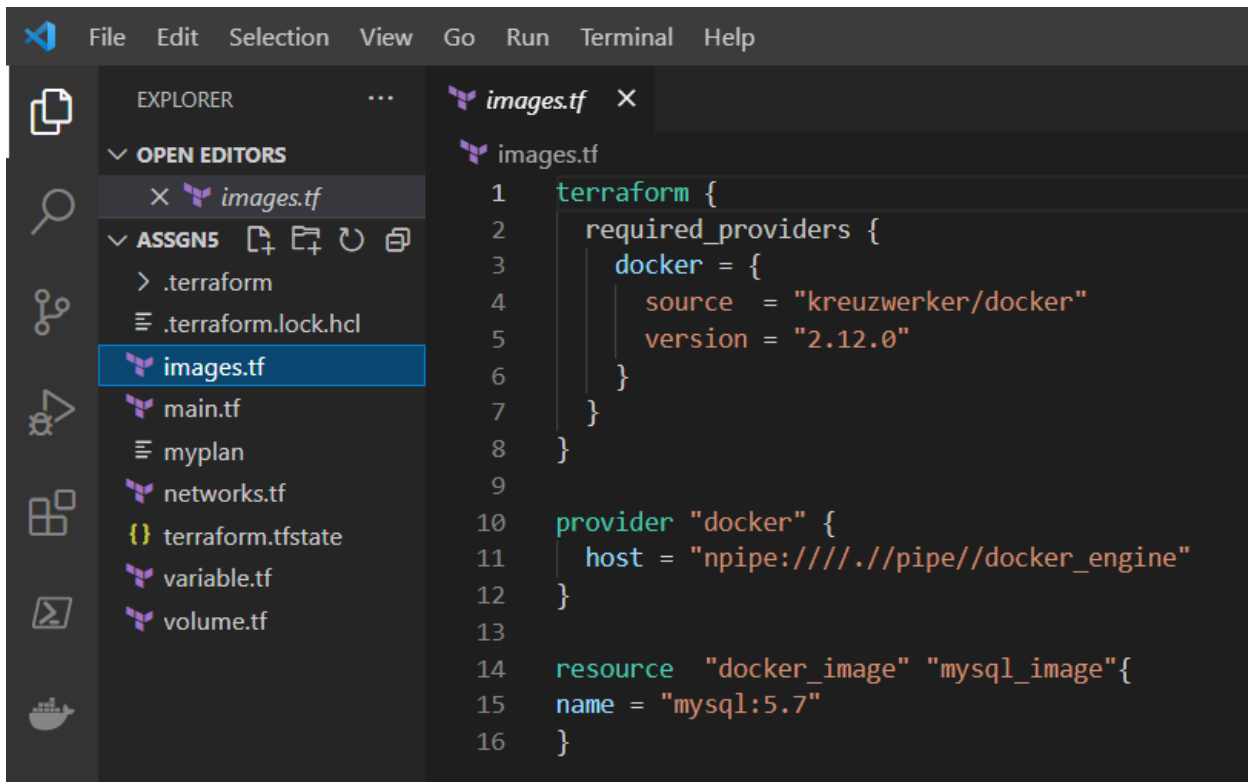
Create the variables file (variables.tf) and add four variables with these default values:

1. container_name: mysql.
2. mysql_root_password: P4sSw0rd0!.
3. mysql_network_name: mysql_internal_network.
4. mysql_volume_name: mysql_data.



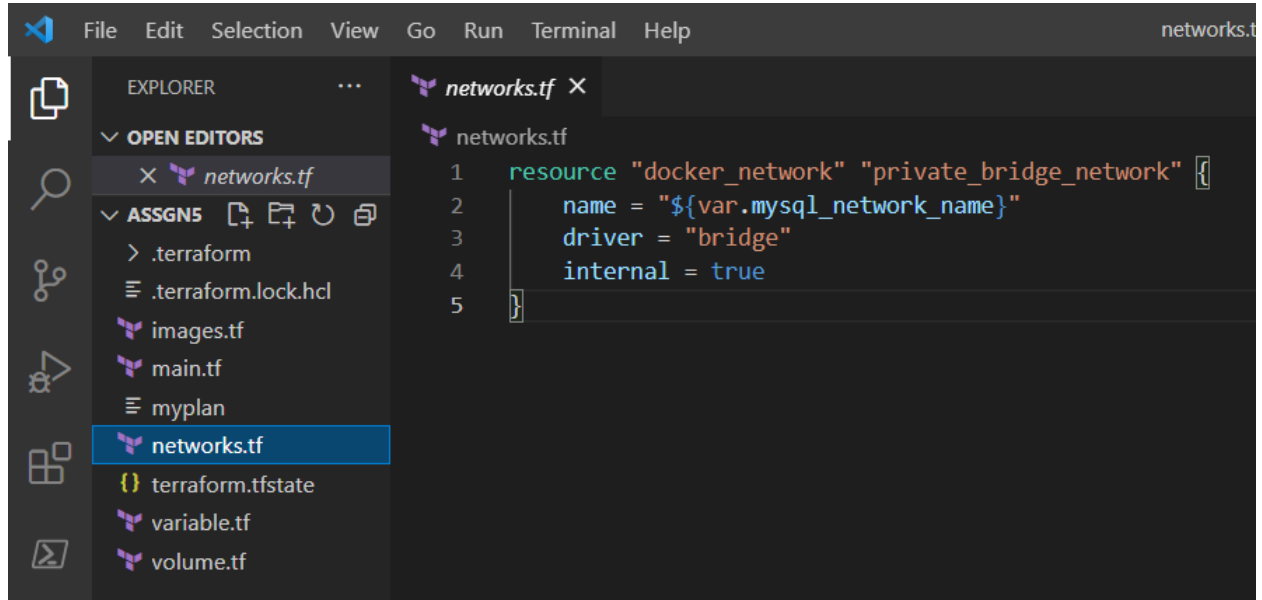
Create the images file (`images.tf`)

1. Add the `docker_image` resource and call it `mysql_image`.
2. Set the name to `mysql:5.7`.



Create the networks file (networks.tf):-

1. Add the docker_network resource and call it private_bridge_network.
2. Set the name to use the mysql_network_name variable.
3. Set the driver to bridge.
4. Set internal to true.

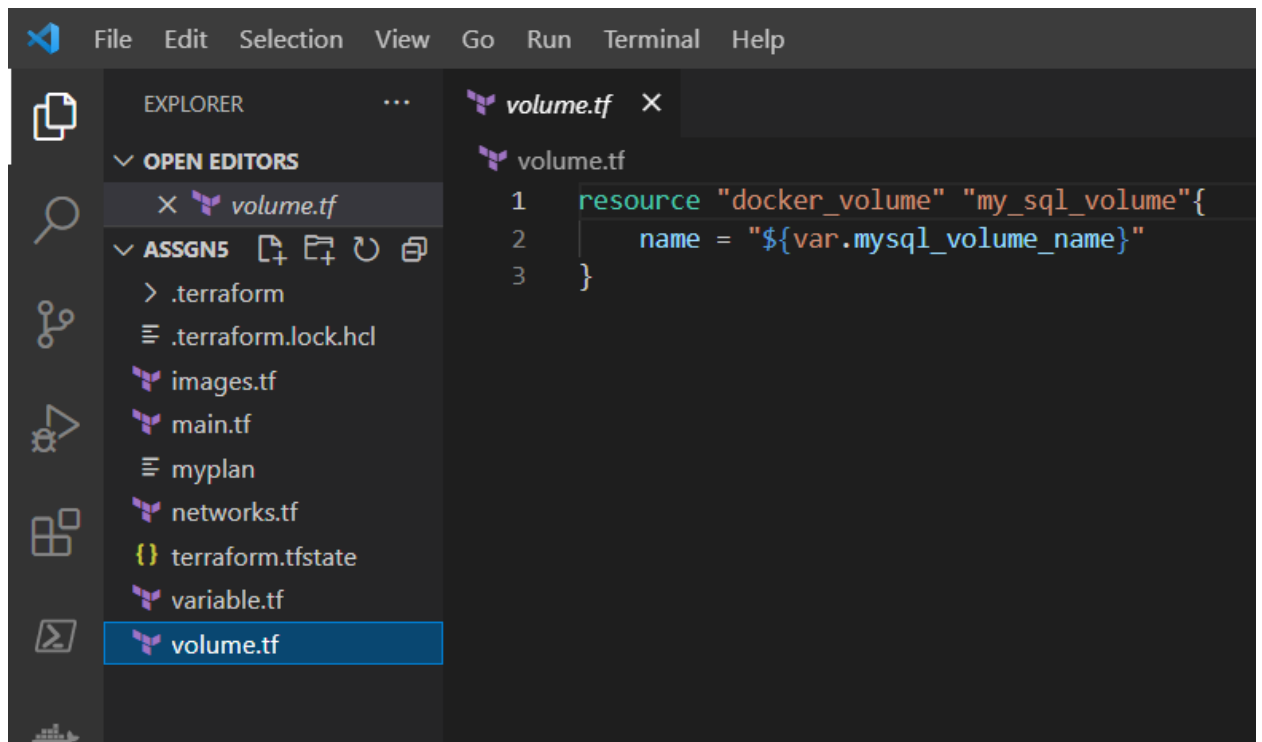


The screenshot shows the Visual Studio Code editor with the 'networks.tf' file open. The Explorer sidebar on the left shows the project structure with files like .terraform, .terraform.lock.hcl, images.tf, main.tf, myplan, networks.tf (selected), terraform.tfstate, variable.tf, and volume.tf. The main editor area displays the following Terraform code:

```
1 resource "docker_network" "private_bridge_network" {  
2     name = "${var.mysql_network_name}"  
3     driver = "bridge"  
4     internal = true  
5 }
```

Create the volumes file (volume.tf):-

1. In volumes.tf add the docker_volume resource and call it mysql_data_volume.
2. Set the name to use the mysql_volume_name variable.

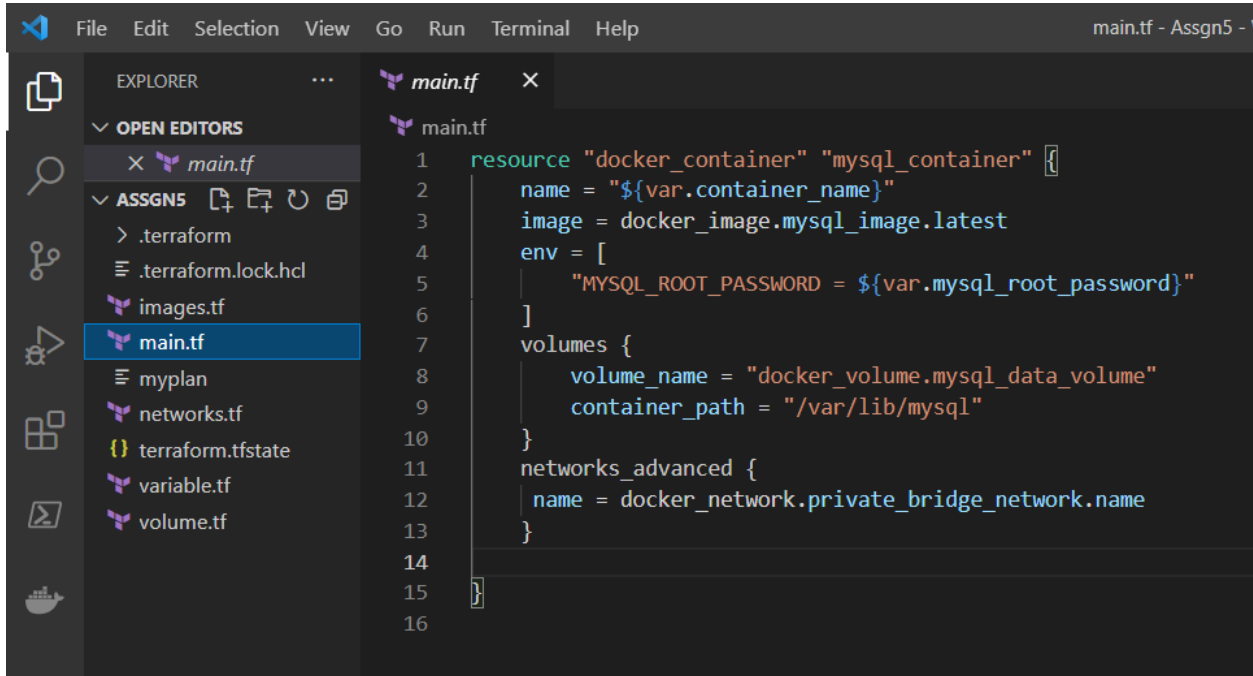


The screenshot shows the Visual Studio Code editor with the 'volume.tf' file open. The Explorer sidebar on the left shows the project structure with files like .terraform, .terraform.lock.hcl, images.tf, main.tf, myplan, networks.tf, terraform.tfstate, variable.tf, and volume.tf (selected). The main editor area displays the following Terraform code:

```
1 resource "docker_volume" "my_sql_volume"{  
2     name = "${var.mysql_volume_name}"  
3 }
```

Create the main file (main.tf):-

1. In main.tf add the docker_container resource and call it mysql_container.
2. Set the name to use the container_name variable.
3. Set the image to use the name of the image coming from docker_image.
4. Create an environment variable for MYSQL_ROOT_PASSWORD and set it to the mysql_root_password variable.
5. Configure the container volume to use the volume created by docker_volume, and make sure the container_path is set to /var/lib/mysql.
7. The container needs to use the network created by docker_network.

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with files like .terraform, .terraform.lock.hcl, images.tf, main.tf (selected), myplan, networks.tf, terraform.tfstate, variable.tf, and volume.tf. The main editor window displays the content of main.tf, which defines a docker_container resource named 'mysql_container'. The resource configuration includes setting the name to a variable, the image to 'docker_image.mysql_image.latest', an environment variable for 'MYSQL_ROOT_PASSWORD', a volume named 'docker_volume.mysql_data_volume' at path '/var/lib/mysql', and an advanced network setting to use 'docker_network.private_bridge_network.name'.

```
1 resource "docker_container" "mysql_container" {  
2   name = "${var.container_name}"  
3   image = docker_image.mysql_image.latest  
4   env = [  
5     "MYSQL_ROOT_PASSWORD = ${var.mysql_root_password}"  
6   ]  
7   volumes {  
8     volume_name = "docker_volume.mysql_data_volume"  
9     container_path = "/var/lib/mysql"  
10  }  
11  networks_advanced {  
12    name = docker_network.private_bridge_network.name  
13  }  
14 }  
15 }  
16 }
```

Deploy the infrastructure

1. Initialize Terraform.
2. Validate the files.
3. Generate a Terraform plan.
4. Deploy the infrastructure using the plan file.

--Commands:

terraform init

terraform validate

terraform plan -out=myplan

terraform apply "myplan"

(used following commands to see network and volume is created)

Docker images

docker ps

docker network ls

docker volume ls

Assignment-6

Using Terraform to Create a RandomID and S3 Buckets

Create the Main file:-

Create the main.tf Terraform file.

Add a provider, aws.

Set the region to use a variable called aws_region.

Add a random_id resource and name it tf_bucket_id.

Set the byte_length to 2.

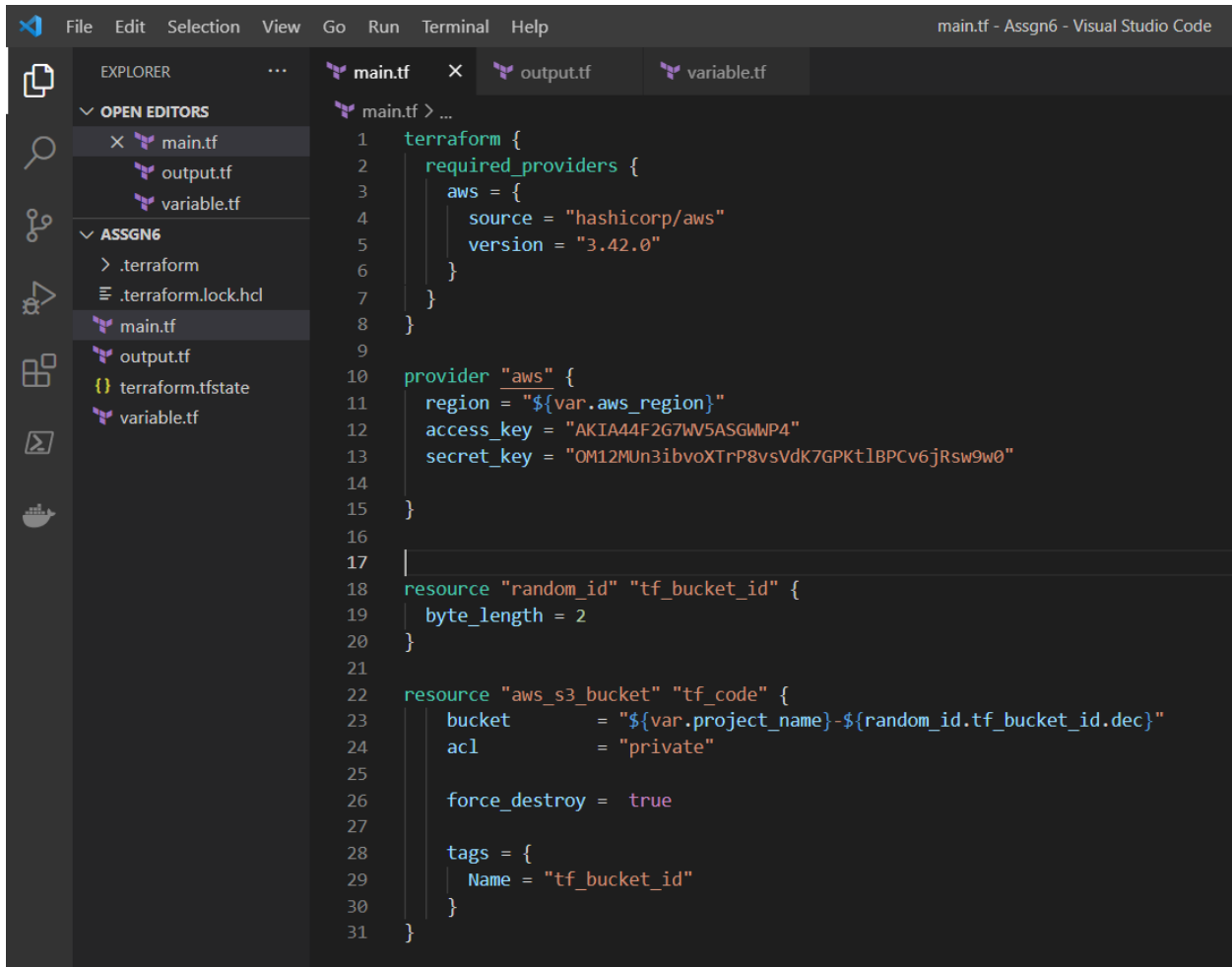
Add a resource, aws_s3_bucket, and name it tf_code.

The bucket name will be set using a variable called project_name, followed by a -, and will use the dec attribute from tf_bucket_id.

Set the acl to private.

Set force_destroy to true.

Create a tag with a name to tf_bucket.



```
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "3.42.0"
6     }
7   }
8 }
9
10 provider "aws" {
11   region = "${var.aws_region}"
12   access_key = "AKIA44F2G7WV5ASGWWP4"
13   secret_key = "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"
14 }
15
16
17
18 resource "random_id" "tf_bucket_id" {
19   byte_length = 2
20 }
21
22 resource "aws_s3_bucket" "tf_code" {
23   bucket      = "${var.project_name}-${random_id.tf_bucket_id.dec}"
24   acl         = "private"
25
26   force_destroy = true
27
28   tags = {
29     Name = "tf_bucket_id"
30   }
31 }
```

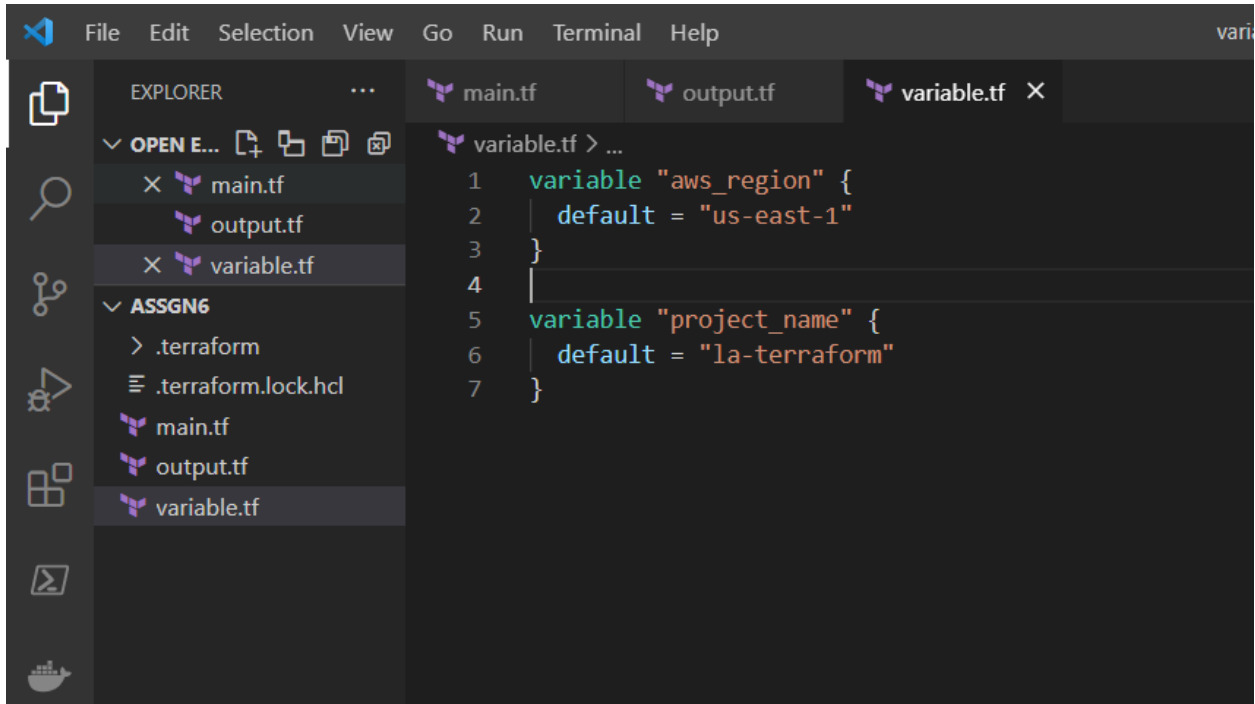
Create the Variables File:-

Create the variables.tf Terraform file.

Add a variable called aws_region.

Set the default to us-east-1. Add a variable called project_name.

Set the default to la-terraform.



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The Explorer shows a project named 'ASSGN6' with files: .terraform, .terraform.lock.hcl, main.tf, output.tf, and variable.tf. The variable.tf file is open in the editor, showing the following Terraform code:

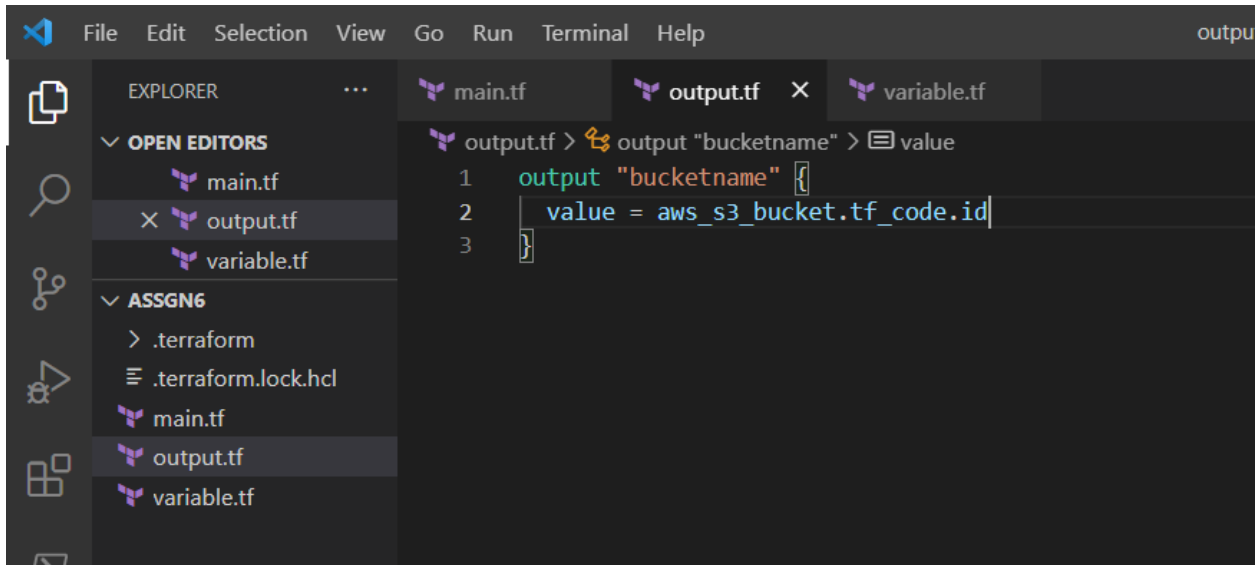
```
1 variable "aws_region" {  
2   | default = "us-east-1"  
3 }  
4  
5 variable "project_name" {  
6   | default = "la-terraform"  
7 }
```

Create the outputs file:-

Create the outputs.tf Terraform file.

Add a output called bucketname.

The value should be set to id, coming from tf_code.



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The Explorer shows the same project 'ASSGN6' with files: .terraform, .terraform.lock.hcl, main.tf, output.tf, and variable.tf. The output.tf file is open in the editor, showing the following Terraform code:

```
1 output "bucketname" {  
2   | value = aws_s3_bucket.tf_code.id  
3 }
```


Deploy the infrastructure:-

Initialize Terraform.

Validate the files.

Deploy the S3 bucket.

terraform init

terraform validate

terraform plan

terraform apply

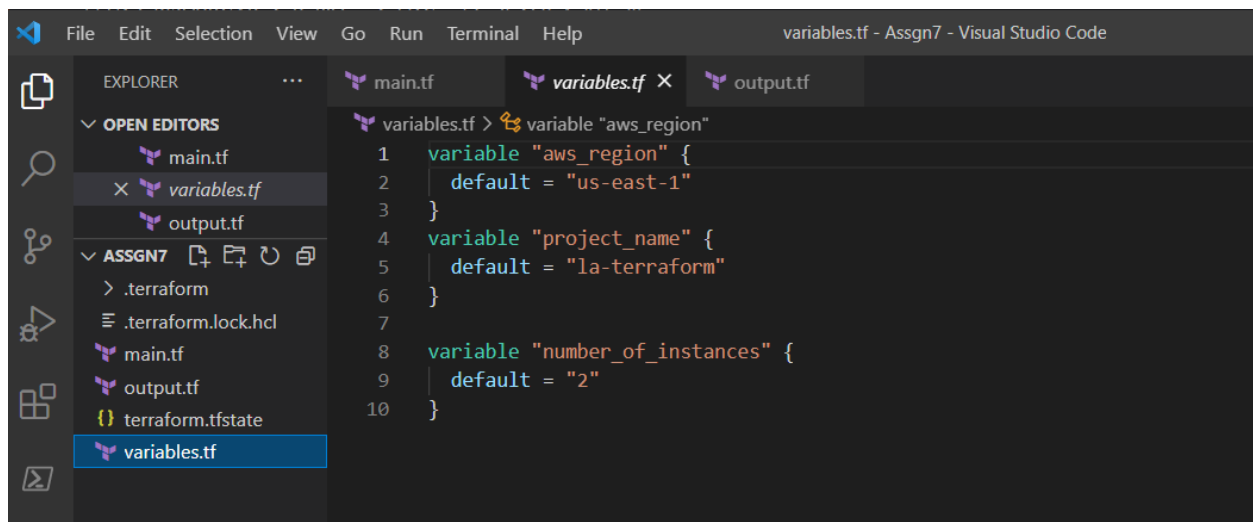
Assignment-7

Update the Variables File:-

Edit variables.tf.

Add a new variable number_of_instances.

Set the the default to 2.



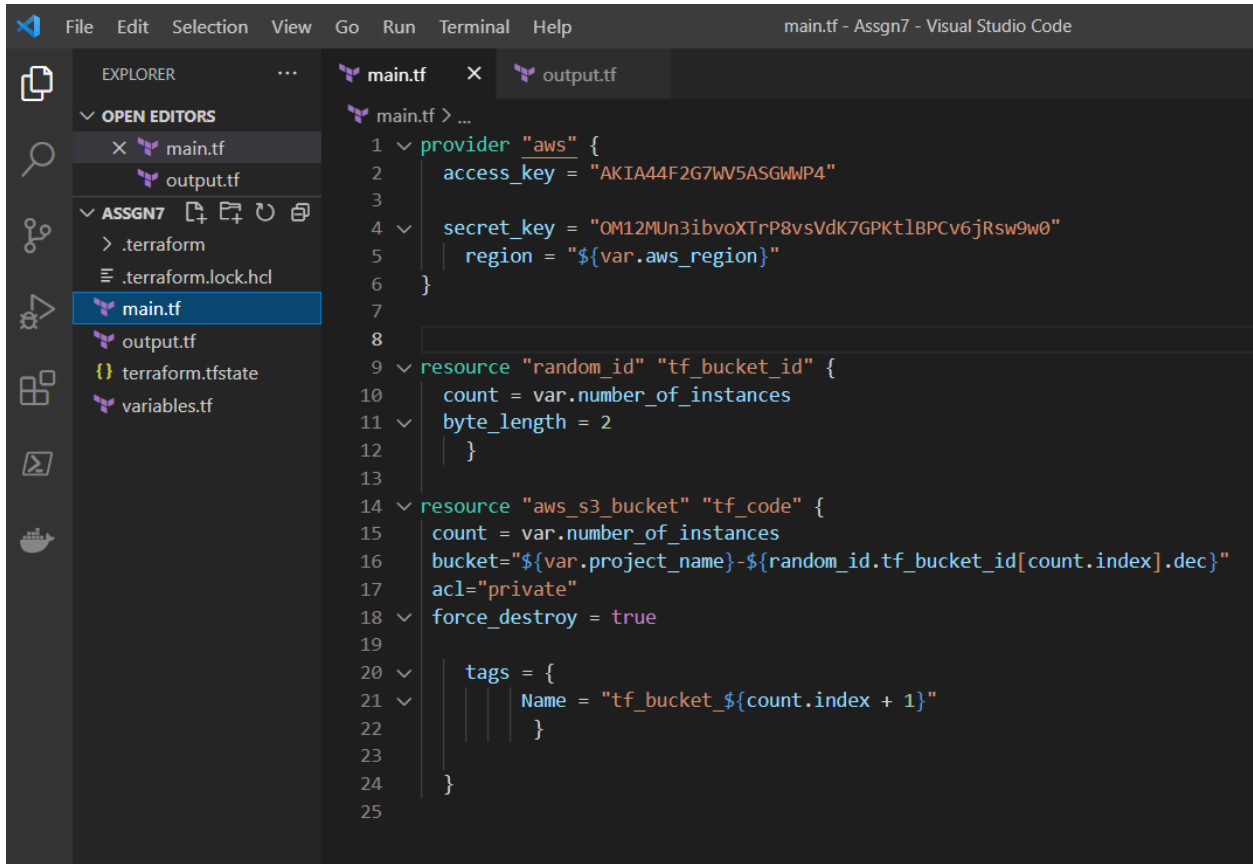
Update the Main File:-

Update random_id and add a count.

Set the value count to use the number_of_instances variable.

Update aws_s3_bucket and add a count.

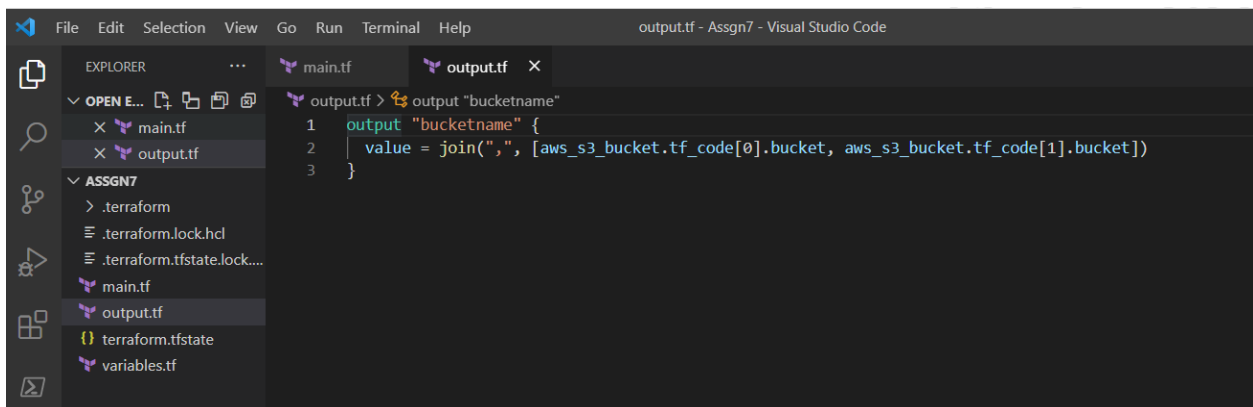
Update random_id.tf_bucket_id.dec so it iterates through the count. Update the Name tag so that tf_bucket is appended with the count index plus one.



```
1 provider "aws" {
2   access_key = "AKIA44F2G7WV5ASGWMP4"
3
4   secret_key = "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"
5   region = "${var.aws_region}"
6 }
7
8
9 resource "random_id" "tf_bucket_id" {
10  count = var.number_of_instances
11  byte_length = 2
12 }
13
14 resource "aws_s3_bucket" "tf_code" {
15  count = var.number_of_instances
16  bucket = "${var.project_name}-${random_id.tf_bucket_id[count.index].dec}"
17  acl = "private"
18  force_destroy = true
19
20  tags = {
21    Name = "tf_bucket_${count.index + 1}"
22  }
23
24 }
25
```

Update the Outputs File:-

Update the bucketname output value to use the join function so that it returns a comma delimited list of bucket names.



```
1 output "bucketname" {
2   value = join(",", [aws_s3_bucket.tf_code[0].bucket, aws_s3_bucket.tf_code[1].bucket])
3 }
```

Deploy the Infrastructure:-

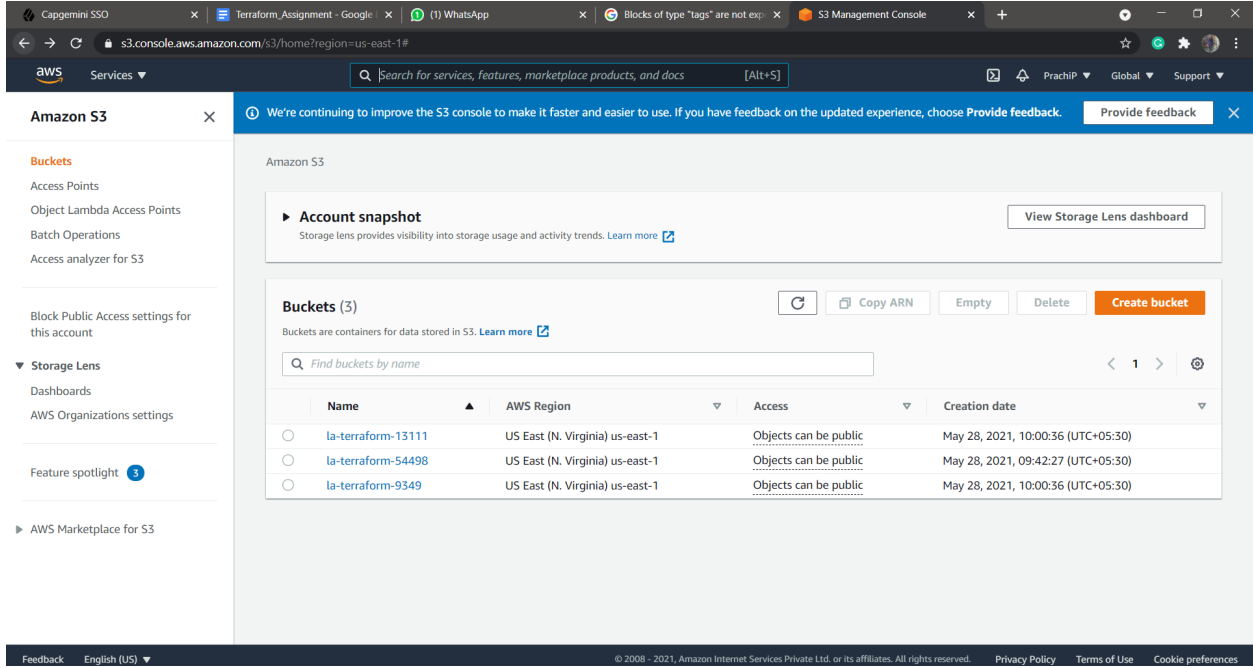
Initialize Terraform.

Validate the files.

Deploy the S3 buckets.

Commands:

Terraform init
Terraform validate
Terraform apply



The screenshot shows the Amazon S3 console interface. The left sidebar contains navigation links for Buckets, Access Points, Object Lambda Access Points, Batch Operations, Access analyzer for S3, Block Public Access settings, Storage Lens, Dashboards, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3. The main content area displays an 'Account snapshot' section with a 'View Storage Lens dashboard' button. Below this is the 'Buckets (3)' section, which includes a search bar and a table of buckets. The table has columns for Name, AWS Region, Access, and Creation date. Three buckets are listed: 'la-terraform-13111', 'la-terraform-54498', and 'la-terraform-9349', all in the 'US East (N. Virginia) us-east-1' region with 'Objects can be public' access.

Amazon S3

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose [Provide feedback](#).

[Provide feedback](#)

Account snapshot
Storage lens provides visibility into storage usage and activity trends. [Learn more](#) [View Storage Lens dashboard](#)

Buckets (3) [Refresh](#) [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

Buckets are containers for data stored in S3. [Learn more](#)

	Name	AWS Region	Access	Creation date
<input type="radio"/>	la-terraform-13111	US East (N. Virginia) us-east-1	Objects can be public	May 28, 2021, 10:00:36 (UTC+05:30)
<input type="radio"/>	la-terraform-54498	US East (N. Virginia) us-east-1	Objects can be public	May 28, 2021, 09:42:27 (UTC+05:30)
<input type="radio"/>	la-terraform-9349	US East (N. Virginia) us-east-1	Objects can be public	May 28, 2021, 10:00:36 (UTC+05:30)

Feedback English (US) © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences