It seems like you're preparing to take the GSP345 challenge lab on Google Cloud Platform using Terraform. This lab involves several tasks that integrate various Google Cloud services and Terraform functionalities. Here’s a step-by-step approach to tackle each task based on the lab instructions:

### Task 1: Create the configuration files

1. \*\*Directory Structure\*\*:

- Create the following directory structure in Cloud Shell:

```

main.tf

variables.tf

modules/

├── instances/

│ ├── instances.tf

│ ├── outputs.tf

│ └── variables.tf

└── storage/

├── storage.tf

├── outputs.tf

└── variables.tf

```

2. \*\*Variables\*\*:

- In `variables.tf` files (root and within modules), define the following variables:

```hcl

variable "region" {

default = "<filled in at lab start>"

}

variable "zone" {

default = ""

}

variable "project\_id" {

default = "<Your Google Cloud Project ID>"

}

```

3. \*\*Provider Configuration\*\*:

- Add the provider block to `main.tf`:

```hcl

provider "google" {

project = var.project\_id

region = var.region

zone = var.zone

}

```

4. \*\*Initialize Terraform\*\*:

- Run the following command in Cloud Shell to initialize Terraform:

```bash

terraform init

```

### Task 2: Import infrastructure

1. \*\*Import Existing Instances\*\*:

- Navigate to Compute Engine > VM Instances in Google Cloud Console.

- Note down details (Instance ID, boot disk image, machine type) of `tf-instance-1` and `tf-instance-2`.

- Update `instances.tf` within `modules/instances/` to define resources for `tf-instance-1` and `tf-instance-2`.

- Example snippet for `instances.tf`:

```hcl

resource "google\_compute\_instance" "tf-instance-1" {

name = "tf-instance-1"

machine\_type = "e2-medium" # Use appropriate machine type

boot\_disk {

initialize\_params {

image = "debian-cloud/debian-10"

}

}

network\_interface {

network = "default"

# Add other necessary configurations

}

}

```

- Import instances using:

```bash

terraform import module.instances.google\_compute\_instance.tf-instance-1 <instance-id>

terraform import module.instances.google\_compute\_instance.tf-instance-2 <instance-id>

```

- Apply changes:

```bash

terraform apply

```

### Task 3: Configure a remote backend

1. \*\*Create Cloud Storage Bucket\*\*:

- Update `storage.tf` in `modules/storage/`:

```hcl

resource "google\_storage\_bucket" "terraform\_state\_bucket" {

name = "<Bucket Name>"

location = "US"

force\_destroy = true

uniform\_bucket\_level\_access = true

}

```

- Initialize and apply changes:

```bash

terraform init

terraform apply

```

2. \*\*Configure Remote Backend\*\*:

- Update `main.tf` to configure Terraform backend:

```hcl

terraform {

backend "gcs" {

bucket = "<Bucket Name>"

prefix = "terraform/state"

}

}

```

- Initialize Terraform and migrate state:

```bash

terraform init -reconfigure

```

### Task 4: Modify and update infrastructure

1. \*\*Update Instance Configurations\*\*:

- Modify `instances.tf` to update `tf-instance-1` and `tf-instance-2` machine types to `e2-standard-2`.

- Add a third instance resource as specified.

2. \*\*Apply Changes\*\*:

- Initialize Terraform and apply changes.

### Task 5: Destroy resources

1. \*\*Remove Third Instance Resource\*\*:

- Remove the third instance resource from configurations.

2. \*\*Apply Changes\*\*:

- Initialize Terraform and apply changes to destroy the third instance.

### Task 6: Use a module from the Registry

1. \*\*Add Network Module\*\*:

- Add the network module from Terraform Registry to `main.tf`.

- Configure with specified parameters.

2. \*\*Update Instances\*\*:

- Update `instances.tf` to connect `tf-instance-1` to `subnet-01` and `tf-instance-2` to `subnet-02`.

### Task 7: Configure a firewall

1. \*\*Create Firewall Rule\*\*:

- Define a firewall rule in `main.tf` to allow TCP port 80 ingress from `VPC Name` network.

2. \*\*Apply Changes\*\*:

- Initialize Terraform and apply changes to create the firewall rule.

By following these steps, you should be able to complete the GSP345 challenge lab successfully. Remember to verify each task after completion to ensure everything is working as expected.