Site Reliability Engineering Test

IT&Digital

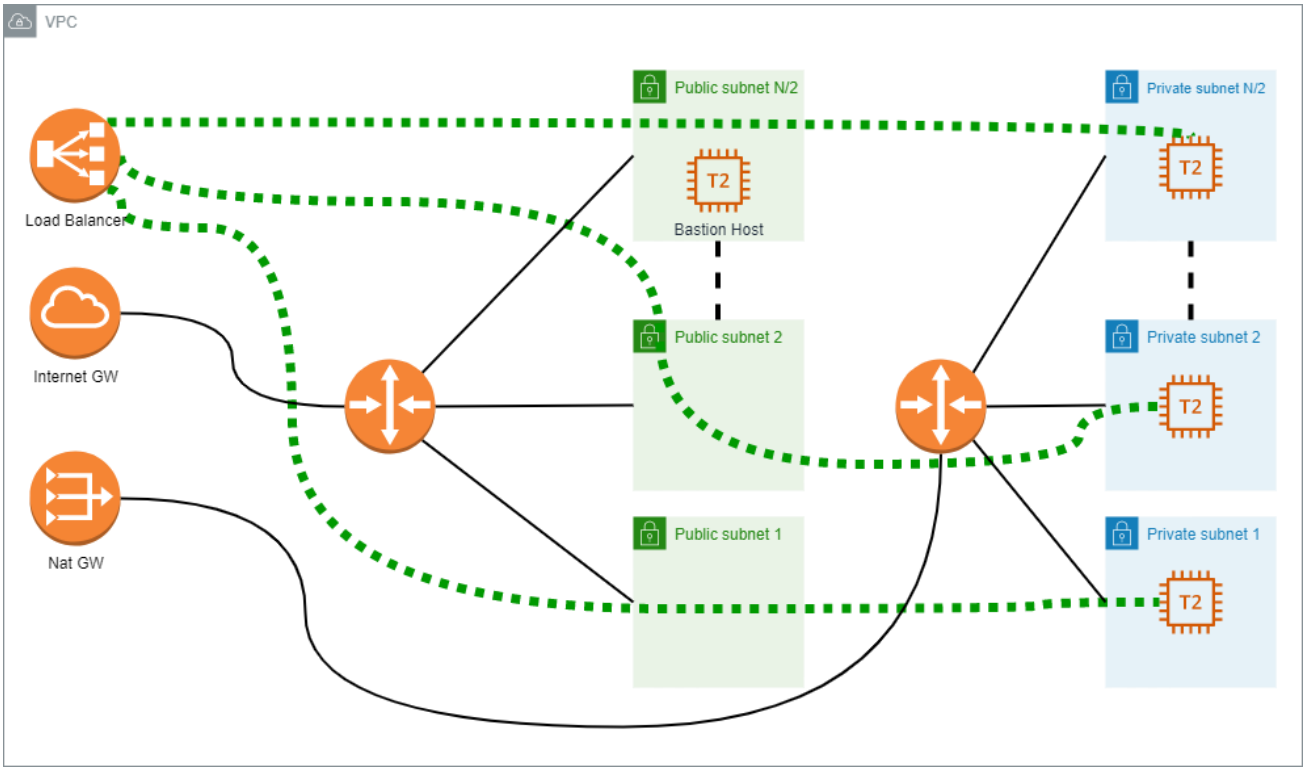
galp

Readme

This document represents a technical test to evaluate the technical capabilities of the candidate necessary for the position to which the candidate apply.
The candidate will have one week to make the exercise from the moment of delivery.
The candidate must develop the code meeting the requirements requested in this exercise and shared it in a public GIT repository (such as GitHub, GitLab or whichever the candidate prefers), sharing the URL with GALP to be able to access and review all the code, as well as to download and execute it.
The code repository must have as many artifacts, codes and documentation (such as any MarkDown file like README.md) as the candidate needs to perform the exercise and that it considers that the reviewer may need for the execution and review of the result.
The code must compile correctly using the "make apply" command and then "make destroy" without design errors in the code or bad coding. If a satisfactory result is not achieved, the review of the exercise will have ended, and the candidate will be automatically discarded.

Description

An Infrastructure as Code (a.k.a IaC) project is requested to be developed over AWS cloud platform such as follows:



Two types of subnets should exist:

- Public subnets. Attached instances and resources have public IP addresses.
- Private subnets. Attached instances and resources don't have public IP addresses.

The total number of private subnets that must exists is N/2, and the total number of public subnets must also be N/2 where N is the maximum number of subnets that can exist in the VPC and N will always be less than or equal to 6.

Each private subnet will have a T2.micro instance attached, and one attached bastion host (also T2.micro) on one of the public subnets (any public subnet). The bastion host (a.k.a. Jump Server) will be used to connect to other instances on private subnets by SSH connection. All instances must use the same SSH key pairs to log-in.
The Internet Outbound for the instances on private subnets will be reach out using a NAT Gateway. The instances on public subnets can connect through a Internet Gateway or NAT Gateway whichever the candidate prefers.
Finally, an Application Load Balancer (a.k.a. ALB) will be used to forward all HTTP traffic to all instances working on private subnets by round robin balancing. However, the ALB will have to be attached on each public subnet.
Once an HTTP connection is attempted to the public DNS created in the creation of this exercise, the following message should be responsed:

"Hello World at <date>"

The <date> clause must be replaced by the date of creation of golden image.

Considerations

1. The resulted exercise will be published on a public GIT repository (such as GitHub, GitLab or whichever the candidate prefers) where a GALP reviewer can clone/download the code to execute and review it.
2. The main code must be coded as a Terraform language, however, the main command interface must be "make" such as GNU Make or similar, coding a Makefile. Other languages are allowed to support the Terraform Code. Ex: A Packer code in HCL language is required below (see Golden_Image module section) and can be called from Makefile or Terraform Code.
3. The exercise must be published in the master/main branch of the public GIT repository.
4. Running a simple "make apply" must create the requested AWS environment described in this document, and a simple "make destroy" must destroy all infrastructure created by the previous command (It is assumed that the reviewer will already be logged into the AWS environment with the correct environment variables).
5. The developed code will require the following input variables (It is assumed that reviewer will aready get and terraform.tfvars file with the input variables set):
   ***Note: Required** means a mandatory variable to be set, but **Optional** means the variable must exist but it is not required to be set*
   a. **Network_CIDR (string, required)** to set the network IP address configuration on CIDR format
   b. **N_Subnets (integer, required)** to set the number of subnets to be used
   c. **Name (string, required)** to set a name to the deployed infrastructure by tag name or another field if required by the created resource.
   d. **Tags (key/value dictionary or map, optional).** A bundle of key/values records (a.k.a tags) to be set on resources that support it.
6. The developed code will response the following outpus once "make apply" was run:
   a. **Private_instances_IP_addresses (Key/Value dictionary or map)**. The private IP address of each attached Instance on the private subnets.
   b. **Bastion_Host_IP (string)**. The public IP address of bastion host.
   c. **SSH_key_content (string)**. The key to be used to SSH any instance.

d. **Load_blanacer_HTTP_Content (string)**. Public DNS name to reach the created HTTP publication.
e. **Usernames (key/value dictionary or map or string).** The usernames to connect the instances (even bastion host) **.** If the username is the same for all instances (even bastion host) then a simple string output variable is enough.

7. The developed code must include the developed terraform modules as described in the “code structure” section.

Code Structure

The developed code can be structured as the candidate prefers (folders, files, etc) but, it is a requirement of the exercise that there are the following modules:

Network module

A module developed in native Terraform language to create the VPC and all required network resources and configurations.

The module will require the following **input variables**:
Note: Required *means a mandatory variable to be set, but* ***Optional*** *means the variable must exist but it is not required to be set.*

- **Network_CIDR (string, required)**. Set the IP address configuration to be configured into VPC and all required network resources.
- **N_subnets (integer, required)**. Set the total subnets to create into VPC. This input variable can only accept an even number, except if the "plus" of this module (see below) is done, then this condition does not apply.
- **Name (string, required)**. To set the name value on Tags or resources field if the resource support or require it (Caution! sometimes the name fields existing in some resources must be unique regarding other resources, thus, it must be handled)
- **Tags (key/value dictionary or map, optional)**. Set a defined tags on the resources that support it.

The module must response the following **output variables**:

- **Network (key/value dictionary or map)**. A dictionary or map with all information regarding the VPC, subnets and any existing settings and resources related to the creation and configuration of networking in this module.

The module must do the following:

- Create a VPC with network settings set in the Network_CIDR variable.
- Create the total subnets set in the N_subnets variable and split the subnets such as:
  - The maximum number of publics subnets must be N/2
  - The maximum number of private subnets must be N/2
- The necessary route tables must be defined to allow connections between subnets as well as Internet
- The Internet outbound for private subnets must be through NAT Gateway

- The Internet outbound for public subnets must be through Internet Gateway or NAT Gateway (choose one)
- The name and/or tags (if set) must be set in all resources that support or require it.

Plus:

- It's a plus if the subnets split were with an odd number of subnets, in such case, the smallest group of subnets will be the public ones and the largest will be the private ones.

Golden_Image module

A module developed in the native Terraform language to manage the resulting manifest file of a packer build after building a golden image to be used for instances on the private subnets using Packer by Hashicorp.

A Golden image must be built using Packer by HashiCorp including the following:

- The packer build will require the following **input variables:**
  - **Name (string,required):** The name of Golden Image to be identified on AWS
- This built must be launched within Makefile and dump a manifest file in json format.
- The Operative System must be GNU/Linux (any distribution open source and free license) using a T2. micro as type of instance.
- The golden image must have an Apache or Nginx installed and listening for 80/TCP and responsing by HTTP the message "Hello World at <date>" which <date> must be replaced by the date of Golden image was created with the format YYYY-MM-DD

The terraform module will require the following **input variables**:
Note: Required *means a mandatory variable to be set, but* ***Optional*** *means the variable must exist but it is not required to be set.*

- **Manifest_path (string, required)**. The path where the manifest file is dumping by packer build command.

The module must response the following **output variables**:

- **Manifest (Key/value dictionary or map).** It must response the manifest file content to be used within instances module.

The module must do the following:

- Once the Packer Golden image is created, the terraform module will import the manifest file as part of the terraform state and, after that, response the content as output variable

Plus:

- It's a plus if the packer build command is launched within same terraform module instead of Makefile.

Instances module

A module developed in the native Terraform language to create the required instances and a bastion host. This module must run after the Network and Golden_Image modules.

The module will require the following **input variables**:
Note: Required *means a mandatory variable to be set, but* ***Optional*** *means the variable must exist but it is not required to be set.*

- **Network (module.Network.output.Network, required)** to set where Instances must be attached.
- **Image (module.Golden_Image.output.Manifest, required)** to set the Golden Image to be used on the attached Instances on private subnets (not for bastion host)
- **Name (string, required)** To set the name value on Tags or resources field if the resource support or require it (Caution! sometimes the name fields existing in some resources must be unique regarding other resources, thus, it must be handled)
- **Tags (key/value dictionary or map, optional)** Set a defined tags on the resources that support it.

The module must response the following **output variables**:

- **Private_Instances_IP_addresses (key/value dictionary or map).** List of private IP addresses of the attached instances on private subnet.
- **Bastion_Host_IP_address (string).** Public IP of bastion host to jump to rest of private Instances.
- **Load_balancer_HTTP_DNS (string)**. Public DNS name to reach the created HTTP publication.
- **SSH_key_Content (string)**. The SSH key to connect all instances (even the bastion host).
- **Usernames (key/value dictionary or map or string).** The usernames to connect the instances (even bastion host)**.** If the username is the same for all instances (even bastion host) then a simple string output variable is enough.

The module must do the following:

- Create an Instance per private subnet set in Network variable of this module.
    - The instances must be created using the created golden image from Golden_Image module set in the manifest variable of this module.
- Create a bastion host using another GNU/Linux image from AWS Marketplace. The chosen image must be free license and open source.
- Create an application load balancer attached to all public subnets and listening for 80/TCP to forward and parsing HTTP requests to 80/TCP in the private instances.
- The instances (including bastion host) must be a T2.micro type.
- Create the SSH key pairs to connect all instances (including the bastion host).
- Create the proper Security Groups to grant the following accesses:
    - **Source:** Public IP where “make” is run. **Destination**: 22/TCP of the bastion host
    - **Source:** Bastion Host instance. **Destination:** 22/TCP of the private Instances
    - **Source:** Any. **Destination:** 80/TCP of the Application Load Balancer
    - **Source:** Application Load Balancer. **Destination**: 80/TCP of the private instances
    - **Allow ICMP from any source to any destination**