Question 3, Part 1 Response

Summary

Terraform code in the 'terraform' folder creates an AWS VPC closely matching what is described in the scenario. Redundant public, private, and database subnets are deployed across two availability zones inside a new VPC. These subnets host a bastion server, application servers, and a Postgres RDS database instance respectively. Security Groups allow communications between hosts and AWS services as necessary to manage or use the test site. All other access is denied.

As the domain, clientapp.com, is a public DNS zone, an AWS Private Hosted Route53 zone was created to resolve the name within the Application Plane VPC. To resolve the name outside of the VPC, a hosts file could be used as a test. Alternatively, bastion1 can be used to resolve the domain and access the test site.

The test site consists of two servers, wpserver1 and wpserver2, hosting a single page listing the name of the responding server. This page serves to prove that load balancing is occurring between the two servers.

Variables are used for most configuration items so that they can be centrally managed.

To deploy the solution:

1. Edit variables.tf to adjust the following:
    a. remote_access_ip: This is the IP that will be allowed to login to the bastion server (with subnet mask).
    b. aws_region: Region the resources should be deployed into.
    c. availability_zones: Availability Zones inside the selected aws_region.
    d. aws_key_name: Name of the key used to access Windows and Linux resources.
2. From within the terraform/production folder issue the commands below:
    a. terraform init
    b. terraform apply
        i. after reviewing the changes, type 'yes'

To destroy the solution:

1. From within the terraform/production folder issue the 'terraform destroy' command.

Files:

variables.tf: Edit values in this file to customize the deployment.

up.js: CloudWatch Synthetics Canary code to monitor the site. Included in clientappUp.zip to be included in terraform canary creation.

clientappUp.zip: Zip file containing up.js used during the creation of the synthetic canary.

user-data.tpl: Template file for customizing Linux guest operating systems.

windows-config.tpl: Template file for customizing Windows guest operating systems.

Issues:

1. The diagram calls for the US-Gov-West-1 region. The solution provides a variable, aws_region, to adjust this value based on the system administrator's existing access.

2. The solution provided does not use SSL certificates. Given that the test domain, clientapp.com, is public, an AWS Certificate Manager Private Certificate Authority would be required to issue certificates. Terraform resource, aws_acmpca_certificate_authority, is available to begin that process. In the interest of time, SSL certificates were left out of the solution.

3. The diagram calls for RDP over TLS 1.2, but does not list a Microsoft RD Gateway server. A gateway server is required to enable RDP over TLS1.2 connectivity. A redundant pair of RD Gateways could be deployed into Public Subnet 1 and Public Subnet 2 to provide this functionality. These were left out in the interest of time. SSL certificates from a trusted CA are also required for this portion of the solution and not available in this lab scenario. As a workaround default RDP access is allowed over TCP/3389.

4. While deploying the Cloudwatch Synthetic Canary to monitor the application, an issue occurred while trying to destroy the resources. Lamba ENIs became 'stuck', preventing the associated subnets, security groups, and VPC from being destroyed. After following the solution provided by AWS, the error below was returned. Given the timing of the exercise, waiting 24 hours was not possible. To avoid the issue from happening again, I have renamed the associated terraform files so that they are not automatically deployed.

   ./findEniAssociations --eni eni-091ab144203d26e49 --region us-east-1.This script is for determining why an ENI that is managed by AWS Lambda has not been deleted.

   Found eni-091ab144203d26e49 with subnet-07c7f154030950fed using Security Groups sg-0ac7f93eeb4aed51c

   Searching for Lambda function versions using subnet-07c7f154030950fed and Security Groups sg-0ac7f93eeb4aed51c...

   No Lambda functions or versions found that were using the same subnet as this ENI.

If this ENI is not deleted automatically in the next 24 hours then it may be 'stuck'. If the ENI will not allow you to delete it manually after 24 hours then please contact AWS support and send them the output of this script.

Assumptions:

Terraform is installed on the deployment workstation.

The AWS CLI is installed and configured for access to the desired AWS resources.

AWS keys exist for logins into Windows and Linux servers.

Resources:

External resources include tool and cloud documentation as well as existing and re-purposed GitHub content.

https://registry.terraform.io/providers/hashicorp/aws/latest/docs
https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/rename-computer?view=powershell-7.2
https://github.com/hashicorp/terraform/issues/1893
https://github.com/guillermo-musumeci/terraform-aws-vpc-3-tier-single-az
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/hosted-zones-private.html
https://www.terraform.io/language/meta-arguments/depends_on
https://troy-ingram.medium.com/terraform-using-variables-and-count-b161e0ce61e1
https://jarombek.com/blog/jul-26-2021-aws-synthetics-canary