**Web Server Infrastructure Provisioning on AWS using Terraform v1.0**

**Document Change Control**

The following table lists the change history for this document. The changes should only be done by authorized person(s) and the respective revision author must update the change in this table for traceability and reference.

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Authors** | **Summary of Changes** |
| 0.1 | July 16, 2022 | Sanjay Kumar | New document. |
|  |  |  |  |
|  |  |  |  |

Contents

[Functional Requirements: 4](#_Toc108902026)

[Proposed Infrastructure Design 5](#_Toc108902027)

[Architecture Components: 6](#_Toc108902028)

[Infrastructure Architecture Design Details 7](#_Toc108902029)

[Assumptions 8](#_Toc108902030)

[Infrastructure affinity for AWS Well-Architected Framework 9](#_Toc108902031)

# Functional Requirements:

**Challenge Scenario**

As the platform/systems team, we have been asked to deliver a standard secure infrastructure on our AWS Cloud Account with following requirements:

Infra Requirements:

* A dedicated VPC with public and private subnets.
* An EC2 in the private subnet running httpd as a service. The timezone of the machine should be set to AEST.
* Download the file belong-test.html from S3 Bucket (belong-coding-challenge) in Sydney region and service it through httpd running as a service on the instance.
* Developers require terminal access to the EC2 for further configuration /administration.

Notes:

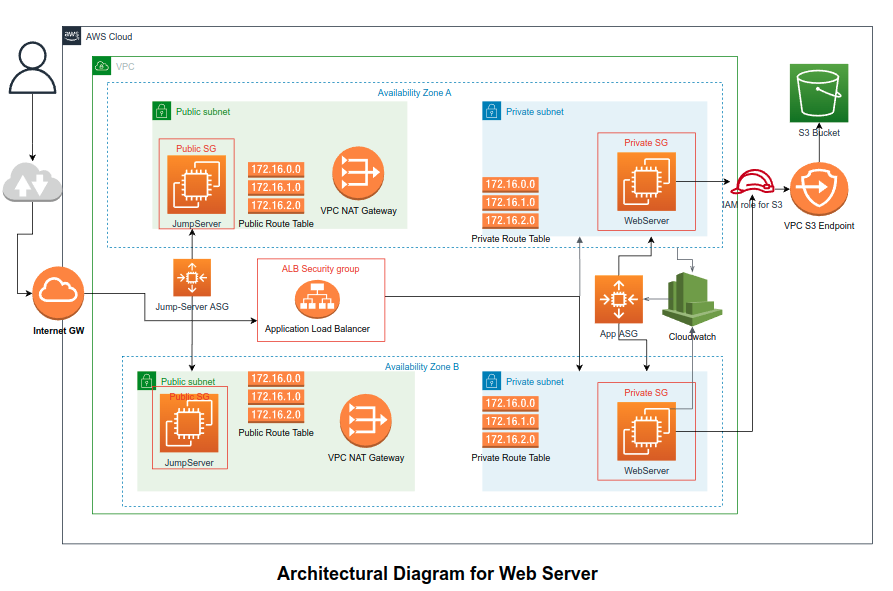
* IaC tool of choice is Cloudformation.
* Solution should be reliable, secure and would soon go into production.

We'll evaluate the submission on your problem-solving approach, simplicity, code quality, and use of Cloud best practices within the solution.

Feel free to be creative and take liberties where you feel it will improve the deliverable!

# Proposed Infrastructure Design

Below is the proposed Architecture for Web Server on AWS Cloud for the above requirements.



# Architecture Components:

Application is designed with below major services/components:

1. VPC (1)
2. Subnets (2 Private Subnets and 2 Public Subnets)
3. Internet Gateway (1)
4. NAT Gateways (1 in each Availability Zone)
5. Route Tables (1 Public Route Table and 2 Private Route Tables)
6. VPC Endpoint (1)
7. Application Load Balancer (1)
8. Auto Scaling Groups (1 for Application Web Server Provisioning and 1 for Jump Server Provisioning)
9. Security Groups (1 for Private Subnet Instances and 1 for Public Subnet Instances , 1 for Application Load Balancer)
   1. ALB Security Group allows only port 80 for HTTP protocol from anywhere.
   2. Public Subnet Security Group allows port 22 for SSH from anywhere.
   3. Private Subnet Security Group Allows port 80 from ALB and port 22 from Public Subnet Security group.

# Infrastructure Architecture Design Details

1. A VPC will created for hosting the application (httpd) service span across two AZs to keep the application highly available. In each AZ, 1 Private and 1 public subnet will be created.
2. Public Subnet will used for creating the resources for internet access. For example, jump server provisioning for accessing the private subnet instances for configuration/administration and NAT Gateway will be provisioned in public subnet for enabling internet access to Private Subnet resources.
3. Private Subnets will be used to provision EC2 instances for hosting the application service. These instances will not be accessible out of the VPC network ensuring greater security; however, they can be accessed using jump servers provisioned in Public Subnet.
4. Internet Gateway will be provisioned for providing internet to the VPC. NAT Gateways will be used to provide internet access to EC2 instances within the private subnet.
5. Private and Public route tables are created for defining the route access inside and outside the VPC network.
6. The EC2 instances for hosting the application will be provisioned via Auto Scaling Group and will span the instances across different AZs. This will help to ensure the application is able to withstand loss of one AZ. The Application ASG will use auto scaling policies for horizontal scaling using CloudWatch alarms.
7. The application will be accessed only through Application Load balancer on http protocol. Even within the VPC.
8. Most of the parameters for Auto Scaling configurations like max size, min size and desired capacity, cool down time, High Threshold, low threshold are configurable and can be defined in the configuration parameter file depending upon the requirements of the application.
9. The terminal access to the EC2 (hosting httpd service) will be through jump server provisioned using Jump-server ASG. The ASG will be capable of provisioning the Jump server in both the AZs to keep the jump server highly available in case of one AZ is down.
10. The S3 bucket will be accessed using VPC Gateway endpoint for enhanced security at a lower price than accessing the bucket from public internet.
11. Current design is configured to run in two availability zones; however, by doing little changes in the variable configuring file, the infrastructure can span in multiple AZs in the same region.

# Assumptions

The source code delivered with this document is developed with below assumptions:

1. The application will be deployed in Sydney (ap-southeast-2) region; however, the region name is configurable from the variable configuration file (terraform.tfvars) using “region” variable.
2. The application will be deployed in two AZs (ap-southeast-2a and ap-southeast-2b) of the Sydney region. AZ's are also configurable from the parameter file terraform.tfvars using “az\_list” variable.
3. The key pair for the application is already created in the aws account and will be configured in the dev-codetest/terraform.tfvars file.
4. The application will be access via application load balancer on port 80.
5. Jump servers will be opened for internet at port 22 for SSH.
6. The S3 Bucket (belong-coding-challenge) and belong-test.html are existing.

# Infrastructure affinity for AWS Well-Architected Framework

1. Operational Excellence
   1. Whole infrastructure is deployed using IaC.
   2. The infrastructure can be easily modifiable by changing attributes in variable configuration file named terraform.tfvars.
2. Security
   1. Web Server is deployed in private subnets, so it is not directly accessible from outside the VPC.
   2. All the instances/load balancers are protected with security groups and allowing only essential ports and protocols.
   3. Application is accessible through load balancer only.
   4. The S3 bucket used for terraform state management is Server Side Encrypted with AES256 algorithm.
3. Reliability
   1. The Web Server is deployed using Auto Scaling Group with MultiAZ. So, the application can sustain failure of up to one AZ.
   2. The Web Server is serving the user traffic via load balancer, which is checking the target health before serving the traffic. So, test recovery procedure is in place for all the instances.
   3. As the application instances are being provisioned via Auto Scaling group with Auto Scaling Policies. So, the application can scale out and scale in horizontally in case of traffic spikes. This auto scaling is configured based on CPU utilization of the instances.
   4. Jump-servers are also provisioned through ASG to keep the jump server available in case of one AZ breakdown.
4. Performance Efficiency
   1. The application can use resources efficiently due to auto scaling policies in place.
   2. The S3 services are being used via VPC Gateway endpoint.
5. Cost Optimization
   1. The S3 traffic is routed through VPC endpoint, so the traffic to S3 is cheaper compared to using it via public internet.
   2. Application will scale in for lower traffic requirements.
   3. Because application is deployed using IaC; so, we can destroy the unnecessary resources after testing is completed successfully. It helps to save significant amount of cost.
   4. Also, the application instance can be changed for the application based on cost and performance from the terraform configuration file. It can be done with almost no downtime.
6. Sustainability
   1. As application is hosted using IaC, so it will be easier to adopt new changes in the application and infrastructure faster than manual configurations.