**Project marks** - 90/100

**MCQ marks**- 10/100

**Deadline**- 24th 9:00 AM AST

**Submission link Google form with MCQ** - <https://docs.google.com/forms/d/e/1FAIpQLSd_th8K5uD94stRg1iMuuudvuwR9uV4u8FmmN-5hm9dFXApHw/viewform>

**Submission form LMS (all days of PDF in one zip file)**- [learn.codingdojo.com/exams](http://learn.codingdojo.com/exams)

**Explanation video-**

**Description:**

Infrastructure as Code service is gaining more traction every day as they offer distinctive benefits for businesses. Your company has hired you in an IT infrastructure management role. Your job is to provide architecture and engineering expertise to partner with your developer team to plan, analyze, design, test, and deploy infrastructure expressed as code.

You have been tasked with writing IaC templates for Cloudformation and Terraform to ensure that technical and operational requirements are achievable and being satisfied with the proposed technologies.

Please provide enough evidence of your work validating and verifying requirements, analysis and final reporting of the actual completion of the work.

in DevOps IaC services refer to a specific approach to

1. Cloudformation
   1. Create a cloudformation stack that will -20
      1. create a custom VPC, two ec2 servers, two security groups
      2. Bootstrap apache2 server
      3. store the application content in the index.html file printing **your name** and **cohort name**.
      4. this application should be publicly accessible.

Paste the template yml.

|  |
| --- |
| AWSTemplateFormatVersion: 2010-09-09  Description: Create a EC2 under a VPC  Resources:    SampleVpc:      Type: AWS::EC2::VPC      Description: Sample VPC      Properties:        CidrBlock: 10.0.0.0/16        Tags:          -            Key: Name            Value: sampleVpc    SampleSubnet:      Type: AWS::EC2::Subnet      Properties:        CidrBlock: 10.0.0.0/24        MapPublicIpOnLaunch: true        VpcId: !Ref SampleVpc    SampleRouteTable:      Type: AWS::EC2::RouteTable      Properties:        VpcId: !Ref SampleVpc    SampleInternetGateway:      Type: AWS::EC2::InternetGateway    SampleGatewayAttachment:      Type: AWS::EC2::VPCGatewayAttachment      Properties:        VpcId: !Ref SampleVpc        InternetGatewayId: !Ref SampleInternetGateway    InternetRoute:      Type: AWS::EC2::Route      DependsOn:        - SampleGatewayAttachment      Properties:        RouteTableId: !Ref SampleRouteTable        GatewayId: !Ref SampleInternetGateway        DestinationCidrBlock: 0.0.0.0/0    SampleSubnetRouteTableAssoc:      Type: AWS::EC2::SubnetRouteTableAssociation      Properties:        RouteTableId: !Ref SampleRouteTable        SubnetId: !Ref SampleSubnet    SampleInstancetest1:      Type: AWS::EC2::Instance      DependsOn:        - InternetRoute        - SampleSubnetRouteTableAssoc      Properties:        InstanceType: t2.micro        SubnetId: !Ref SampleSubnet        ImageId: ami-065efef2c739d613b        SecurityGroupIds:          - !Ref SampleSecurityGroup          - !Ref SampleSecurityGrouptest        UserData:          Fn::Base64:            !Sub |              #!/bin/bash              sudo sudo              yum update -y              yum install httpd -y              systemctl start httpd              systemctl enable httpd      SampleInstancetest2:      Type: AWS::EC2::Instance      DependsOn:        - InternetRoute        - SampleSubnetRouteTableAssoc      Properties:        InstanceType: t2.micro        SubnetId: !Ref SampleSubnet        ImageId: ami-065efef2c739d613b        SecurityGroupIds:          - !Ref SampleSecurityGroup          - !Ref SampleSecurityGrouptest        UserData:          Fn::Base64:            !Sub |              #!/bin/bash              sudo sudo              yum update -y              yum install httpd -y              systemctl start httpd              systemctl enable httpd    SampleSecurityGroup:      Type: AWS::EC2::SecurityGroup      Properties:        GroupDescription: Sample security group        VpcId: !Ref SampleVpc        SecurityGroupIngress:          -            CidrIp: 0.0.0.0/0            IpProtocol: tcp            FromPort: 22            ToPort: 22      SampleSecurityGrouptest:      Type: AWS::EC2::SecurityGroup      Properties:        GroupDescription: Sample security group        VpcId: !Ref SampleVpc        SecurityGroupIngress:          -            CidrIp: 0.0.0.0/0            IpProtocol: tcp            FromPort: 80            ToPort: 80 |
| Write the commands one by one used to launch the template  aws cloudformation create-stack --stack-name vpctest1 --template-body file://vpc.yaml |
| Web capture_25-6-2022_6027_us-east-1.console.aws.amazon.com (1) |
| Web capture_25-6-2022_6254_3.90.184.142 |
| Web capture_25-6-2022_6721_us-east-1.console.aws.amazon.com |
| Web capture_25-6-2022_61122_us-east-1.console.aws.amazon.com |

* 1. Create a cloudformation with - 25
     1. allowed instances 1-5
     2. Default IP 0.0.0.0 and allow only standard IP format.
     3. Allow only t2.nano, t2.micro, t2.small, t2.medium instances
     4. Public port 80 access but port 22 access only from your own IP
     5. One instances of amazon linux Latest AMI of amazon linux and one instance of ubuntu taken as parameter
     6. User proper tagging with instance name - YOURNAME-cloudformation
     7. Output AZ, DNS, public IP

Paste template yml.

|  |
| --- |
| AWSTemplateFormatVersion: 2010-09-09  Description: Create a EC2  Parameters:    StanderdIPFormat:        Description : The IP address range that can be used to SSH to the EC2 instances        Type: String        MinLength: '9'        MaxLength: '18'        Default: 0.0.0.0/0        AllowedPattern: '(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})'        ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x    InstanceTypeParameter:      Type: String      Default: t2.micro      AllowedValues:        - t2.micro        - t2.small        - t2.medium        - t2.nano      Description: Enter t2.micro, t2.small, t2.medium, or t2.nano. Default is t2.micro.  Id:    Type: AWS::EC2::Image::Id    Default:  ami-065efef2c739d613b    Description: Enter any AMI as Parameters  Resources:    SampleInstancetest:      Type: AWS::EC2::Instance      Properties:        InstanceType:          Ref: InstanceTypeParameter        ImageId: !Ref 'Id'        SecurityGroupIds:          - !Ref SampleSecurityGroup        Tags :          - Key: name            Value: alrashedi-cloudformation    SampleSecurityGroup:      Type: AWS::EC2::SecurityGroup      Properties:        GroupDescription: Sample security group        SecurityGroupIngress:          - IpProtocol: tcp            FromPort: 22            ToPort: 22            CidrIp: !Ref StanderdIPFormat          - IpProtocol: tcp            FromPort: 80            ToPort: 80            CidrIp: 0.0.0.0/0  Outputs:    PublicIP:      Description: Public ip address of the newly created EC2 instance      Value: !GetAtt [SampleInstancetest, PublicIp]    AvailabilityZone:     Description: AvailabilityZone of the newly created EC2 instance     Value: !GetAtt [SampleInstancetest, AvailabilityZone]    PublicAddress:      Description: PublicDnsName of the newly created EC2 instance      Value: !GetAtt [SampleInstancetest, PublicDnsName] |
| Write the commands one by one used to launch the template :  Aws cloudformation create-stack --stack-name mohammed --template-body  <File://ec2.yaml>parameters  parameterKey=instanceTypeparameter.parameterValue=t2.nano |

1. Terraform
   1. Dojo jump is going to be launched soon. It aims to deploy it in Apache Servers. You and your colleagues have started to work on the project. Your Teammate have developed the website and they need your help to build infrastructure for deploying the website. - 20  
        
      Dojo-jump game link - <https://github.com/chandradeoarya/dojo-jump>
      1. Ubuntu or amazon linux in server
      2. Proper ingress and egress
      3. Proper tagging

Paste all the template yml files like output, variable, main etc.

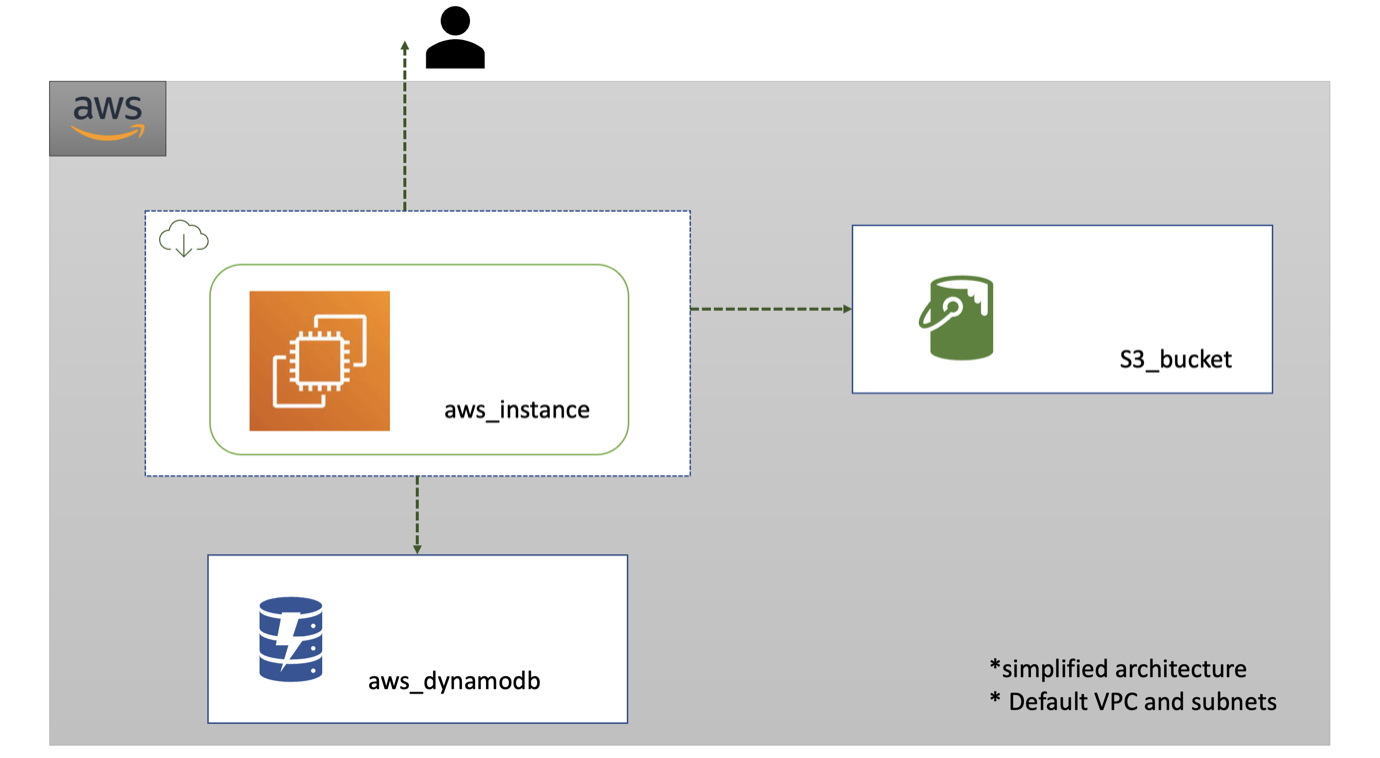
|  |
| --- |
| Install.sh  #!/bin/sh  sudo su  yum update -y  yum install -y httpd.x86\_64  chmod -R 777 /var/www/html  cd /var/www/html  wget https://raw.githubusercontent.com/chandradeoarya/dojo-jump/master/style.css  wget https://raw.githubusercontent.com/chandradeoarya/dojo-jump/master/main.js  wget https://raw.githubusercontent.com/chandradeoarya/dojo-jump/master/index.html  systemctl start httpd.service  systemctl enable httpd.service  Main.tf  resource "aws\_key\_pair" "default" {  key\_name = "key"  public\_key = file("${var.key\_path}")  }  # Define the security group  resource "aws\_security\_group" "sgweb" {  name = "DojoJump-ml"  description = "Allow incoming HTTP connections & SSH access"  ingress {  description = "Allow incoming HTTP connections"  from\_port = 80  to\_port = 80  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  }  ingress {  description = "Allow incoming SSH access"  from\_port = 22  to\_port = 22  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  }  egress {  from\_port = 0  to\_port = 0  protocol = "-1"  cidr\_blocks = ["0.0.0.0/0"]  }  tags = {  Name = "DojoJump mo"  }  }  # Define an Amazon Linux instance with Apache web server  resource "aws\_instance" "DojoJump" {  ami = var.amazon\_linux\_ami  instance\_type = var.instance\_type  key\_name = aws\_key\_pair.default.id  vpc\_security\_group\_ids = ["${aws\_security\_group.sgweb.id}"]  user\_data = file("install.sh")  tags = {  Name = "DojoJumpmo"  }  }  Provider.tf  terraform {  required\_providers {  aws = {  source = "hashicorp/aws"  version = "4.18.0"  }  }  }  # Define AWS as a provider  provider "aws" {  region = var.aws\_region  }  ############################################################  Variables.tf  variable "aws\_region" {  description = "Region for the EC2"  default = "us-east-1"  }  variable "amazon\_linux\_ami" {  description = "Amazon linux AMI for EC2"  default = "ami-0cff7528ff583bf9a"  }  variable "instance\_type" {  description = "instance type"  default = "t2.micro"  }  variable "key\_path" {  description = "SSH Public Key path"  default = "key.pub"  }  Output.tf  output "instance\_id\_DojoJump" {  description = "Instance ID"  value = aws\_instance.DojoJump.id  }  output "instance\_public\_ip\_DojoJump" {  description = "Instance Public IP"  value = aws\_instance.DojoJump.public\_ip  }  output "instance\_AZ\_name\_DojoJump" {  description = "availability zone"  value = aws\_instance.DojoJump.availability\_zone  } |
| Write the commands one by one used to launch the template .  Terraform init  Terraform plan  Terraform apply |

* 1. Terraform custom module - 25

Let us consider that an organization “CodingDojo” has a blueprint of a prototype of an application “Payroll” that needs to deploy at several countries.

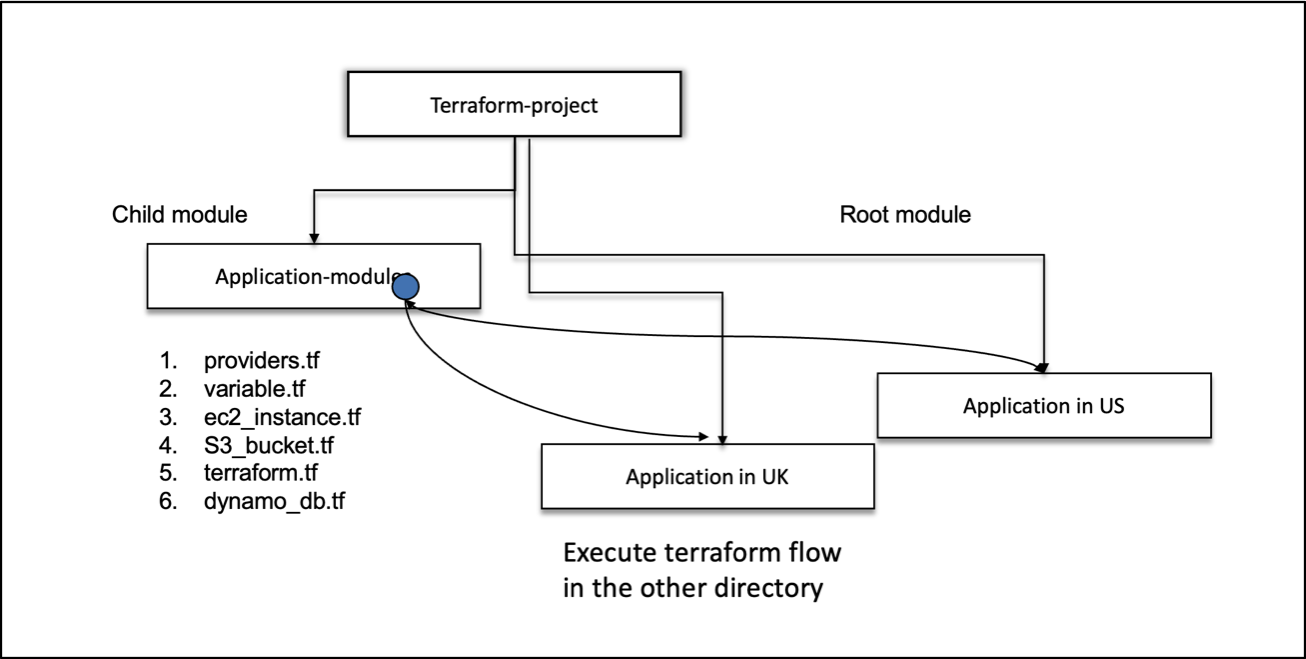
Each country will have its own instance of software deployed on an AWS instance using the same architecture.

Architecture design –



**Important checkpoints:**

1. It consists of a single ec2instance using the custom AMI, that holds the application server.
2. A DynamoDB NoSQL database that will be used to store the data of the employees
3. S3 bucket which will be used to save tax and other documents.
4. Users can access the application through this EC2 instance.
5. This is the architecture of the application in most simplified form.
6. Default VPC and subnets are used.



Create a custom module to deploy the whole application infrastructure.

Use this module to deploy the application in us-east-1. Paste the screenshot of the resources created.

Paste all the template yml files like output, variable, main etc.

|  |
| --- |
|  |
| Write the commands one by one used to launch the template. |