**DESCRIPTION**

You are hired as a cloud architect in a global media company. You have been asked to set up a WordPress instance to publish blogs for your company per the defined specifications.

**Background of the problem statement:**

Your organization publishes blogs and provides documentation services for other businesses and technologies. You have been asked to:

* Set up a live WordPress instance to publish blogs
* Set up a WordPress instance that can be used for development and testing purposes so that any work done on this instance will not impact the live blog
* Configure the WordPress instance for development and testing purposes, which will be available only for the business hours (9 AM–6 PM)
* Give access of the WordPress instance to the blogging team for development and testing purposes

**You must use the following:**

* Amazon CloudFormation
* AWS Auto Scaling
* Amazon Machine Images
* Amazon Identity and Access Management

**The following requirements should be met:**

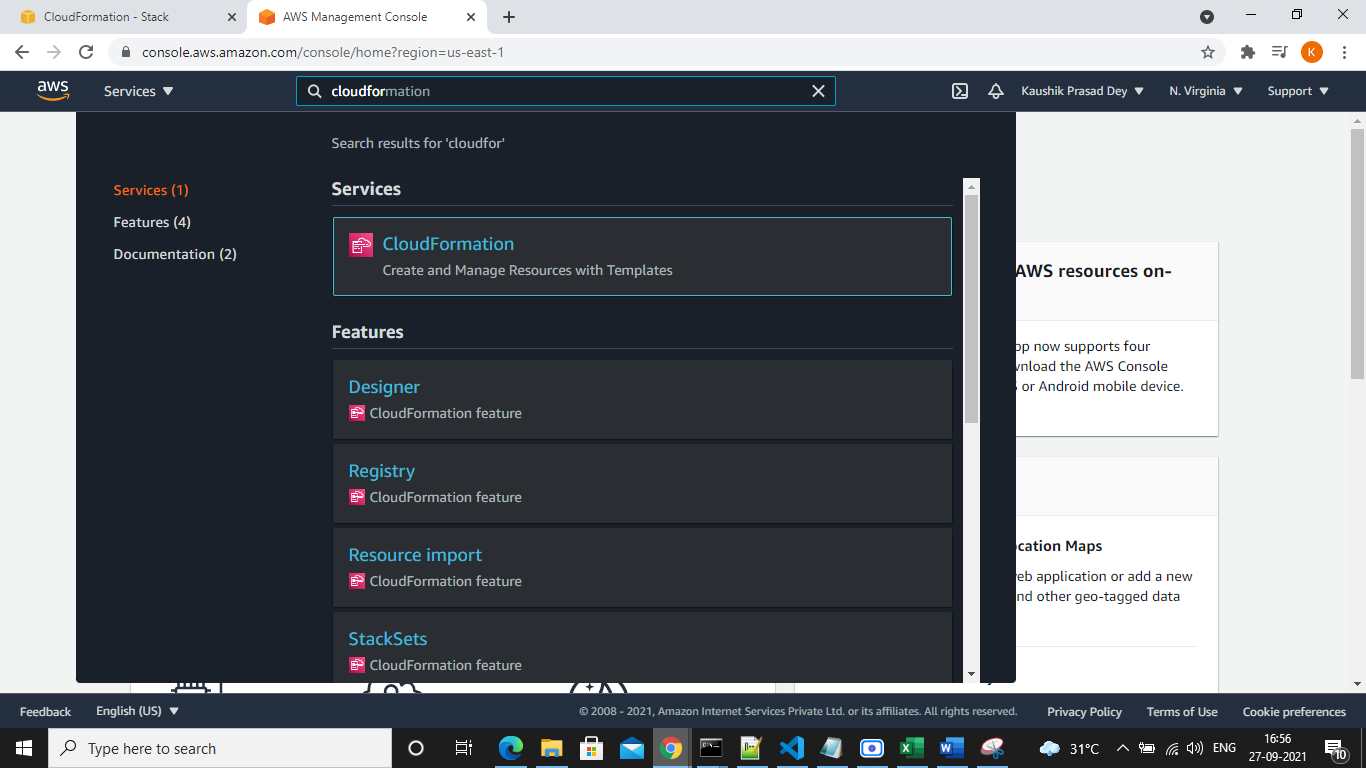
* Ensure that the live WordPress instance has public read access from anywhere over the internet
* Ensure that you block the public access to the WordPress instance for development and testing purposes
* Ensure that only the blogging team has access to the WordPress instance for development and testing purposes
* Ensure that the WordPress instance for development and testing purposes gets shut down after 6 PM every day

Graphical user interface, application, website

Description automatically generated

A screenshot of a computer

Description automatically generated



Mappings

Conditions

Parameters

Description

Resources

Outputs

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

View in Designer Tab

{

    "AWSTemplateFormatVersion": "2010-09-09",

    "Description": "This Template Creates a LAMP Stack for wordpress development.",

    "Parameters": {

        "KeyName": {

            "Description": "Key Pair name",

            "Type": "AWS::EC2::KeyPair::KeyName",

            "Default": "mykey"

        }

    },

    "Mappings": {

        "EC2RegionMap": {

            "ap-northeast-1": {"AmazonLinuxAMIHVMEBSBacked64bit": "ami-cbf90ecb"},

            "ap-south-1": {"AmazonLinuxAMIHVMEBSBacked64bit": "ami-f9daac96"},

            "ap-south": {"AmazonLinuxAMIHVMEBSBacked64bit": "ami-f9daac96"},

            "eu-central-1": {"AmazonLinuxAMIHVMEBSBacked64bit": "ami-a8221fb5"},

            "eu-west-1": {"AmazonLinuxAMIHVMEBSBacked64bit": "ami-a10897d6"},

            "sa-east-1": {"AmazonLinuxAMIHVMEBSBacked64bit": "ami-b52890a8"},

            "us-east-1": {"AmazonLinuxAMIHVMEBSBacked64bit": "ami-1ecae776"},

            "us-west-1": {"AmazonLinuxAMIHVMEBSBacked64bit": "ami-d114f295"},

            "us-west-2": {"AmazonLinuxAMIHVMEBSBacked64bit": "ami-e7527ed7"}

        }

    },

    "Resources": {

        "VPC": {

            "Type": "AWS::EC2::VPC",

            "Properties": {

                "CidrBlock": "172.31.0.0/16",

                "EnableDnsHostnames": "true"

            }

        },

        "InternetGateway": {

            "Type": "AWS::EC2::InternetGateway",

            "Properties": {

            }

        },

        "VPCGatewayAttachment": {

            "Type": "AWS::EC2::VPCGatewayAttachment",

            "Properties": {

                "VpcId": {"Ref": "VPC"},

                "InternetGatewayId": {"Ref": "InternetGateway"}

            }

        },

        "SubnetA": {

            "Type": "AWS::EC2::Subnet",

            "Properties": {

                "AvailabilityZone": {"Fn::Select": ["0", {"Fn::GetAZs": ""}]},

                "CidrBlock": "172.31.38.0/24",

                "VpcId": {"Ref": "VPC"}

            }

        },

        "SubnetB": {

            "Type": "AWS::EC2::Subnet",

            "Properties": {

                "AvailabilityZone": {"Fn::Select": ["1", {"Fn::GetAZs": ""}]},

                "CidrBlock": "172.31.37.0/24",

                "VpcId": {"Ref": "VPC"}

            }

        },

        "RouteTable": {

            "Type": "AWS::EC2::RouteTable",

            "Properties": {

                "VpcId": {"Ref": "VPC"}

            }

        },

        "RouteTableAssociationA": {

            "Type": "AWS::EC2::SubnetRouteTableAssociation",

            "Properties": {

                "SubnetId": {"Ref": "SubnetA"},

                "RouteTableId": {"Ref": "RouteTable"}

            }

        },

        "RouteTableAssociationB": {

            "Type": "AWS::EC2::SubnetRouteTableAssociation",

            "Properties": {

                "SubnetId": {"Ref": "SubnetB"},

                "RouteTableId": {"Ref": "RouteTable"}

            }

        },

        "RoutePublicNATToInternet": {

            "Type": "AWS::EC2::Route",

            "Properties": {

                "RouteTableId": {"Ref": "RouteTable"},

                "DestinationCidrBlock": "0.0.0.0/0",

                "GatewayId": {"Ref": "InternetGateway"}

            },

            "DependsOn": "VPCGatewayAttachment"

        },

        "NetworkAcl": {

            "Type": "AWS::EC2::NetworkAcl",

            "Properties": {

                "VpcId": {"Ref": "VPC"}

            }

        },

        "SubnetNetworkAclAssociationA": {

            "Type": "AWS::EC2::SubnetNetworkAclAssociation",

            "Properties": {

                "SubnetId": {"Ref": "SubnetA"},

                "NetworkAclId": {"Ref": "NetworkAcl"}

            }

        },

        "SubnetNetworkAclAssociationB": {

            "Type": "AWS::EC2::SubnetNetworkAclAssociation",

            "Properties": {

                "SubnetId": {"Ref": "SubnetB"},

                "NetworkAclId": {"Ref": "NetworkAcl"}

            }

        },

        "NetworkAclEntryIngress": {

            "Type": "AWS::EC2::NetworkAclEntry",

            "Properties": {

                "NetworkAclId": {"Ref": "NetworkAcl"},

                "RuleNumber": "100",

                "Protocol": "-1",

                "RuleAction": "allow",

                "Egress": "false",

                "CidrBlock": "0.0.0.0/0"

            }

        },

        "NetworkAclEntryEgress": {

            "Type": "AWS::EC2::NetworkAclEntry",

            "Properties": {

                "NetworkAclId": {"Ref": "NetworkAcl"},

                "RuleNumber": "100",

                "Protocol": "-1",

                "RuleAction": "allow",

                "Egress": "true",

                "CidrBlock": "0.0.0.0/0"

            }

        },

        "LoadBalancer": {

            "Type": "AWS::ElasticLoadBalancing::LoadBalancer",

            "Properties": {

                "Subnets": [{"Ref": "SubnetA"}, {"Ref": "SubnetB"}],

                "LoadBalancerName": "aws-simplearn-elb",

                "Listeners": [{

                    "InstancePort": "80",

                    "InstanceProtocol": "HTTP",

                    "LoadBalancerPort": "80",

                    "Protocol": "HTTP"

                }],

                "HealthCheck": {

                    "HealthyThreshold": "2",

                    "Interval": "5",

                    "Target": "TCP:80",

                    "Timeout": "3",

                    "UnhealthyThreshold": "2"

                },

                "SecurityGroups": [{"Ref": "LoadBalancerSecurityGroup"}],

                "Scheme": "internet-facing"

            },

            "DependsOn": "VPCGatewayAttachment"

        },

        "LoadBalancerSecurityGroup": {

            "Type": "AWS::EC2::SecurityGroup",

            "Properties": {

                "GroupDescription": "aws-simplearn-elb-sg",

                "VpcId": {"Ref": "VPC"},

                "SecurityGroupIngress": [{

                    "CidrIp": "0.0.0.0/0",

                    "FromPort": 80,

                    "IpProtocol": "tcp",

                    "ToPort": 80

                }]

            }

        },

        "WebServerSecurityGroup": {

            "Type": "AWS::EC2::SecurityGroup",

            "Properties": {

                "GroupDescription": "aws-simplearn-sg",

                "VpcId": {"Ref": "VPC"},

                "SecurityGroupIngress": [{

                    "CidrIp": "0.0.0.0/0",

                    "FromPort": 22,

                    "IpProtocol": "tcp",

                    "ToPort": 22

                }, {

                    "FromPort": 80,

                    "IpProtocol": "tcp",

                    "SourceSecurityGroupId": {"Ref": "LoadBalancerSecurityGroup"},

                    "ToPort": 80

                }]

            }

        },

        "DatabaseSecurityGroup": {

            "Type": "AWS::EC2::SecurityGroup",

            "Properties": {

                "GroupDescription": "aws-simplearn-db-sg",

                "VpcId": {"Ref": "VPC"},

                "SecurityGroupIngress": [{

                    "IpProtocol": "tcp",

                    "FromPort": "3306",

                    "ToPort": "3306",

                    "SourceSecurityGroupId": {"Ref": "WebServerSecurityGroup"}

                }]

            }

        },

        "Database": {

            "Type": "AWS::RDS::DBInstance",

            "Properties": {

                "AllocatedStorage": "5",

                "BackupRetentionPeriod": "0",

                "DBInstanceClass": "db.t2.micro",

                "DBInstanceIdentifier": "aws-simplearn-db",

                "DBName": "wordpress",

                "Engine": "MySQL",

                "MasterUsername": "wordpress",

                "MasterUserPassword": "wordpress",

                "VPCSecurityGroups": [{"Fn::GetAtt": ["DatabaseSecurityGroup", "GroupId"]}],

                "DBSubnetGroupName": {"Ref": "DBSubnetGroup"}

            },

            "DependsOn": "VPCGatewayAttachment"

        },

        "DBSubnetGroup" : {

            "Type" : "AWS::RDS::DBSubnetGroup",

            "Properties" : {

                "DBSubnetGroupDescription" : "DB subnet group",

                "SubnetIds": [{"Ref": "SubnetA"}, {"Ref": "SubnetB"}]

            }

        },

        "LaunchConfiguration": {

            "Type": "AWS::AutoScaling::LaunchConfiguration",

            "Metadata": {

                "AWS::CloudFormation::Init": {

                    "config": {

                        "packages": {

                            "yum": {

                                "php": [],

                                "php-mysql": [],

                                "mysql": [],

                                "httpd": []

                            }

                        },

                        "sources": {

                            "/var/www/html": "https://wordpress.org/wordpress-4.7.2.tar.gz"

                        },

                        "files": {

                            "/tmp/config": {

                                "content": {"Fn::Join": ["", [

                                    "#!/bin/bash -ex\n",

                                    "cp /var/www/html/wordpress/wp-config-sample.php /var/www/html/wordpress/wp-config.php\n",

                                    "sed -i \"s/'database\_name\_here'/'wordpress'/g\" wp-config.php\n",

                                    "sed -i \"s/'username\_here'/'wordpress'/g\" wp-config.php\n",

                                    "sed -i \"s/'password\_here'/'wordpress'/g\" wp-config.php\n",

                                    "sed -i \"s/'localhost'/'", {"Fn::GetAtt": ["Database", "Endpoint.Address"]}, "'/g\" wp-config.php\n",

                                    "chmod -R 777 wp-content/ \n"

                                ]]},

                                "mode": "000500",

                                "owner": "root",

                                "group": "root"

                            }

                        },

                        "commands": {

                            "01\_config": {

                                "command": "/tmp/config",

                                "cwd": "/var/www/html/wordpress"

                            }

                        },

                        "services": {

                            "sysvinit": {

                                "httpd": {

                                    "enabled": "true",

                                    "ensureRunning": "true"

                                }

                            }

                        }

                    }

                }

            },

            "Properties": {

                "EbsOptimized": false,

                "ImageId": {"Fn::FindInMap": ["EC2RegionMap", {"Ref": "AWS::Region"}, "AmazonLinuxAMIHVMEBSBacked64bit"]},

                "InstanceType": "t2.micro",

                "SecurityGroups": [{"Ref": "WebServerSecurityGroup"}],

                "KeyName": {"Ref": "KeyName"},

                "AssociatePublicIpAddress": true,

                "UserData": {"Fn::Base64": {"Fn::Join": ["", [

                    "#!/bin/bash -ex\n",

                    "yum update -y aws-cfn-bootstrap\n",

                    "/opt/aws/bin/cfn-init -v --stack ", {"Ref": "AWS::StackName"}, " --resource LaunchConfiguration --region ", {"Ref": "AWS::Region"}, "\n",

                    "/opt/aws/bin/cfn-signal -e $? --stack ", {"Ref": "AWS::StackName"}, " --resource AutoScalingGroup --region ", {"Ref": "AWS::Region"}, "\n"

                ]]}}

            }

        },

        "AutoScalingGroup": {

            "Type": "AWS::AutoScaling::AutoScalingGroup",

            "Properties": {

                "LoadBalancerNames": [{"Ref": "LoadBalancer"}],

                "LaunchConfigurationName": {"Ref": "LaunchConfiguration"},

                "MinSize": "2",

                "MaxSize": "2",

                "DesiredCapacity": "2",

                "VPCZoneIdentifier": [{"Ref": "SubnetA"}, {"Ref": "SubnetB"}]

            },

            "CreationPolicy": {

                "ResourceSignal": {

                    "Timeout": "PT10M"

                }

            },

            "DependsOn": "VPCGatewayAttachment"

        }

    },

    "Outputs": {

        "URL": {

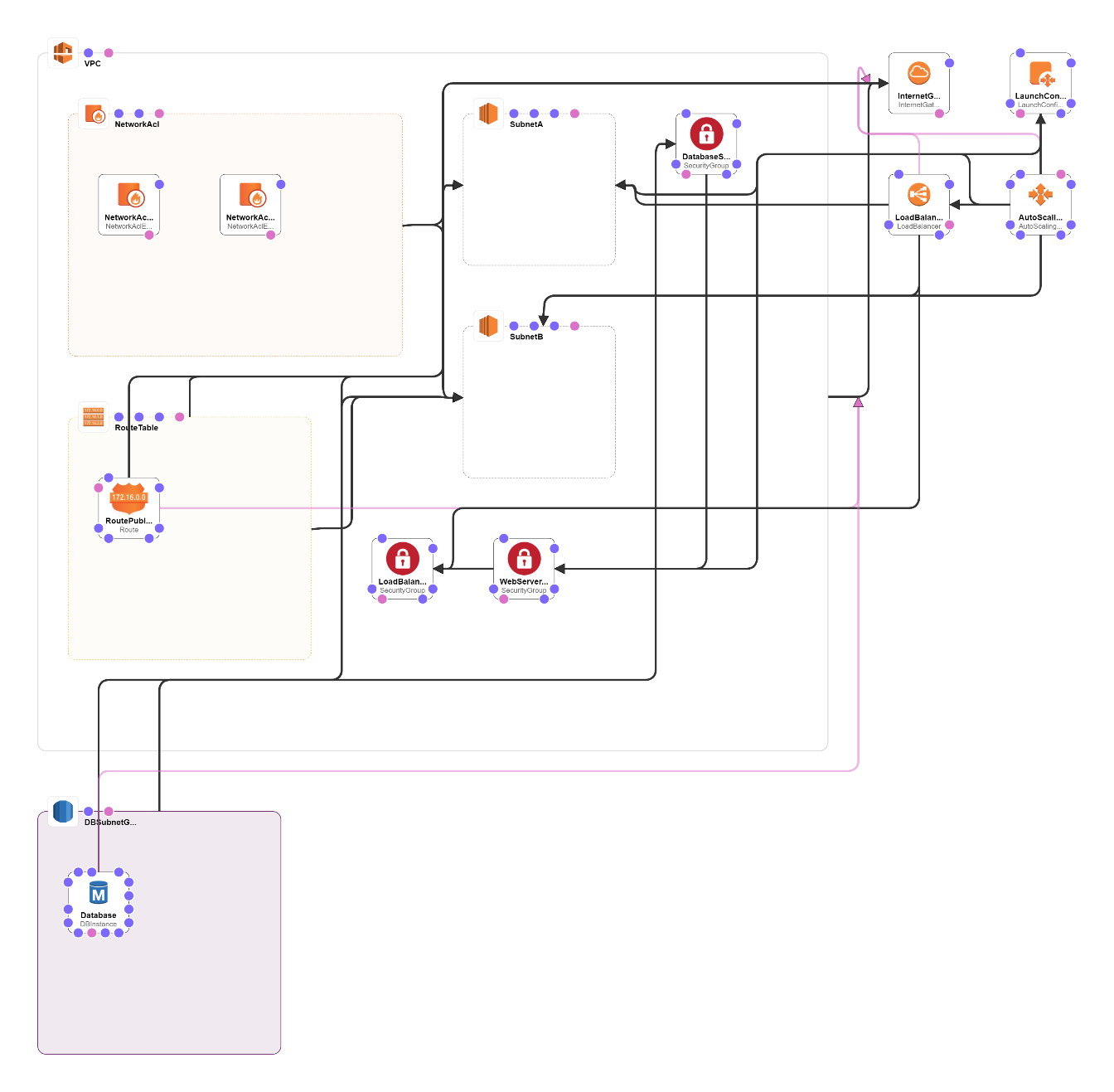
            "Value": {"Fn::Join": ["", ["http://", {"Fn::GetAtt": ["LoadBalancer", "DNSName"]}, "/wordpress"]]},

            "Description": "Wordpress URL"

        }

    }

}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

# **Architecture:**

Diagram

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

wget <https://s3.amazonaws.com/solutions-reference/aws-instance-scheduler/latest/scheduler-cli.zip>

unzip scheduler-cli.zip

cd scheduler-cli

python setup.py install

scheduler-cli create-period --stack Ec2instanceScheduler --region us-east-1 --name firstdayofmonth --begintime 09:00 --endtime 06:00 --monthdays 1

scheduler-cli create-schedule --stack Ec2instanceScheduler --name dayone --region us-east-1 --periods firstdayofmonth --timezone UTC

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated