**Hello,**

**Below are the questions for the DevOps Assignment. You will have 24 hours to complete the assignment and send it back to us. This will help us evaluate your candidature for the role.**

**Question 1**

Write shell commands to do the below:

1. Kill all processes/zombie processes of service called “gunicorn” in a single command.

**killall gunicorn**

1. MySQL shell command to show the unique IPs from where MySQL connections are being made to the Database.

**show processlist;**

1. Bash command to get value of version number of 3 decimal points (first occurrence) from a file containing the JSON:

{

"name": "abc",

"version": "1.0",

"version": "1.0.57",

"description": "Testing",

"main": "src/server/index.js",

"version": "1.1"

}

**cat <”filename”> | awk 'NR==3'**

1. Bash command to add these numbers from a file and find average upto 2 decimal points:

0.0238063905753

0.0308368914424

0.0230014918637

0.0274232220275

0.0184563749986

**awk '{ SUM += $1; count++ } END { print SUM/count }' file.txt**

**Question 2**

Write an executable ​**bash**​script to set up a whole LAMP stack, PHP app can be Wordpress and DB can be MySQL.

The script should meet the below requirements

* This script should install all components needed for a Wordpress website.
* We should be able to run this script on a local machine or server and after the execution of the script, it should have Wordpress Running via Nginx/Apache.
* A database user for Wordpress should also be made automatically from within the script and same should be set in Wordpress conf file. The script should output the database user details at the end of a successful installation as a MySQL connection string.

#!/bin/sh

# Update packages and Upgrade system

echo -e Updating System.

sudo apt-get update -y && sudo apt-get upgrade -y

## Install AMP

echo -e Installing Apache2

sudo apt-get install apache2 apache2-doc apache2-mpm-prefork apache2-utils libexpat1 ssl-cert -y

echo -e Installing PHP & Requirements

sudo apt-get install libapache2-mod-php5 php5 php5-common php5-curl php5-dev php5-gd php5-idn php-pear php5-imagick php5-mcrypt php5-mysql php5-ps php5-pspell php5-recode php5-xsl -y

echo -e Installing MySQL

sudo apt-get install mysql-server mysql-client libmysqlclient15.dev -y

echo -e Installing phpMyAdmin

sudo apt-get install phpmyadmin -y

echo -e Verifying installs

sudo apt-get install apache2 libapache2-mod-php5 php5 mysql-server php-pear php5-mysql mysql-client mysql-server php5-mysql php5-gd -y

# Permissions

echo -e Permissions for /var/www

sudo chown -R www-data:www-data /var/www

# Enabling Mod Rewrite, required for WordPress permalinks and .htaccess files

echo -e Enabling Modules

sudo a2enmod rewrite

sudo php5enmod mcrypt

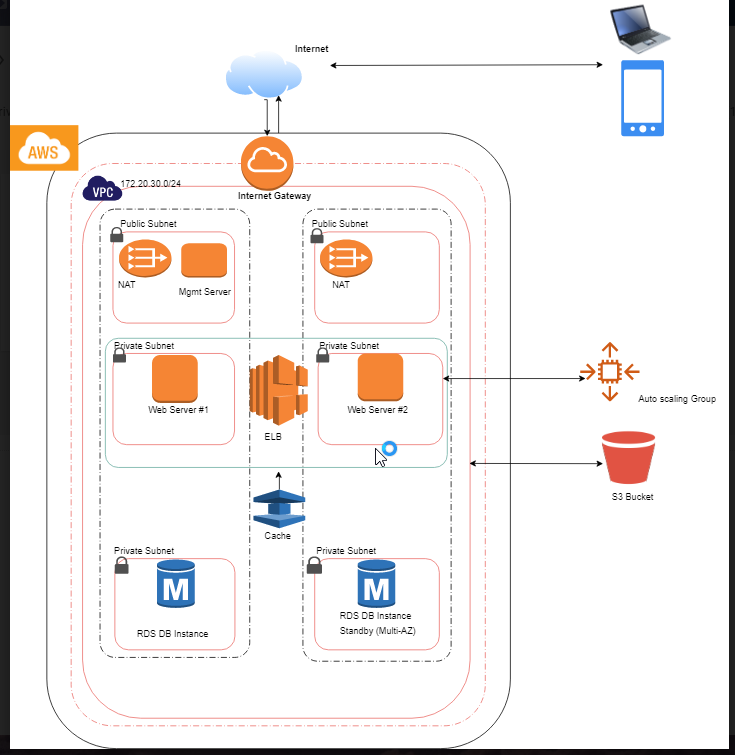
# Restart Apache

echo -e Restarting Apache

sudo service apache2 restart

**Question 3**

Let’s say you are working on an application which is hosted on AWS or Azure. Draw an architecture diagram for a PHP/JAVA/Python-based application to be hosted on AWS with all mentions like VPC, AWS/any other cloud platform services, well-defined network segregation. Any more details that you think are necessary please do include them.



**BONUS QUESTIONS**

1. Write a script which will ​**based on “Number of requests”** ​metric of the ALB/ELB scale up web-app EC2 instances under the Load Balancer, increase AWS Elasticsearch Nodes count, and change the instance size of a MongoDB EC2 instance from m4.large to m4.xlarge. (without using ASG) (Can be done for any cloud platform).

* aws autoscaling update-auto-scaling-group --auto-scaling-group-name **Elasticsearch -AS-Group** --min-size 4 --max-size 10 --desired-capacity 4

Instance should be on stop state before modifying.

* aws ec2 modify-instance-attribute \ --instance-id **Mongodb-Instance-id** \ --instance-type "{\"Value\": \"m4.xlarge\"}"

1. Write a Terraform/Cloud Formation template for the LAMP stack in Question 2.

{

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

"Description" : "AWS CloudFormation Sample Template.",

"Parameters" : {

"KeyName": {

"Description" : "Name of an existing EC2 KeyPair to enable SSH access to the instance",

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

"ConstraintDescription" : "must be the name of an existing EC2 KeyPair."

},

"DBName": {

"Default": "MyDatabase",

"Description" : "MySQL database name",

"Type": "String",

"MinLength": "1",

"MaxLength": "64",

"AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]\*",

"ConstraintDescription" : "must begin with a letter and contain only alphanumeric characters."

},

"DBUser": {

"NoEcho": "true",

"Description" : "Username for MySQL database access",

"Type": "String",

"MinLength": "1",

"MaxLength": "16",

"AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]\*",

"ConstraintDescription" : "must begin with a letter and contain only alphanumeric characters."

},

"DBPassword": {

"NoEcho": "true",

"Description" : "Password for MySQL database access",

"Type": "String",

"MinLength": "1",

"MaxLength": "41",

"AllowedPattern" : "[a-zA-Z0-9]\*",

"ConstraintDescription" : "must contain only alphanumeric characters."

},

"DBRootPassword": {

"NoEcho": "true",

"Description" : "Root password for MySQL",

"Type": "String",

"MinLength": "1",

"MaxLength": "41",

"AllowedPattern" : "[a-zA-Z0-9]\*",

"ConstraintDescription" : "must contain only alphanumeric characters."

},

"InstanceType" : {

"Description" : "WebServer EC2 instance type",

"Type" : "String",

"Default" : "t2.small",

"AllowedValues" : [ "t1.micro", "t2.nano", "t2.micro", "t2.small", "t2.medium", "t2.large", "m1.small", "m1.medium", "m1.large", "m1.xlarge", "m2.xlarge", "m2.2xlarge", "m2.4xlarge", "m3.medium", "m3.large", "m3.xlarge", "m3.2xlarge", "m4.large", "m4.xlarge", "m4.2xlarge", "m4.4xlarge", "m4.10xlarge", "c1.medium", "c1.xlarge", "c3.large", "c3.xlarge", "c3.2xlarge", "c3.4xlarge", "c3.8xlarge", "c4.large", "c4.xlarge", "c4.2xlarge", "c4.4xlarge", "c4.8xlarge", "g2.2xlarge", "g2.8xlarge", "r3.large", "r3.xlarge", "r3.2xlarge", "r3.4xlarge", "r3.8xlarge", "i2.xlarge", "i2.2xlarge", "i2.4xlarge", "i2.8xlarge", "d2.xlarge", "d2.2xlarge", "d2.4xlarge", "d2.8xlarge", "hi1.4xlarge", "hs1.8xlarge", "cr1.8xlarge", "cc2.8xlarge", "cg1.4xlarge"]

,

"ConstraintDescription" : "must be a valid EC2 instance type."

},

"SSHLocation" : {

"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."

}

},

"Mappings" : {

"AWSInstanceType2Arch" : {

"t1.micro" : { "Arch" : "PV64" },

"t2.nano" : { "Arch" : "HVM64" },

"t2.micro" : { "Arch" : "HVM64" },

"t2.small" : { "Arch" : "HVM64" },

"t2.medium" : { "Arch" : "HVM64" },

"t2.large" : { "Arch" : "HVM64" },

"m1.small" : { "Arch" : "PV64" },

"m1.medium" : { "Arch" : "PV64" },

"m1.large" : { "Arch" : "PV64" },

"m1.xlarge" : { "Arch" : "PV64" },

"m2.xlarge" : { "Arch" : "PV64" },

"m2.2xlarge" : { "Arch" : "PV64" },

"m2.4xlarge" : { "Arch" : "PV64" },

"m3.medium" : { "Arch" : "HVM64" },

"m3.large" : { "Arch" : "HVM64" },

"m3.xlarge" : { "Arch" : "HVM64" },

"m3.2xlarge" : { "Arch" : "HVM64" },

"m4.large" : { "Arch" : "HVM64" },

"m4.xlarge" : { "Arch" : "HVM64" },

"m4.2xlarge" : { "Arch" : "HVM64" },

"m4.4xlarge" : { "Arch" : "HVM64" },

"m4.10xlarge" : { "Arch" : "HVM64" },

"c1.medium" : { "Arch" : "PV64" },

"c1.xlarge" : { "Arch" : "PV64" },

"c3.large" : { "Arch" : "HVM64" },

"c3.xlarge" : { "Arch" : "HVM64" },

"c3.2xlarge" : { "Arch" : "HVM64" },

"c3.4xlarge" : { "Arch" : "HVM64" },

"c3.8xlarge" : { "Arch" : "HVM64" },

"c4.large" : { "Arch" : "HVM64" },

"c4.xlarge" : { "Arch" : "HVM64" },

"c4.2xlarge" : { "Arch" : "HVM64" },

"c4.4xlarge" : { "Arch" : "HVM64" },

"c4.8xlarge" : { "Arch" : "HVM64" },

"g2.2xlarge" : { "Arch" : "HVMG2" },

"g2.8xlarge" : { "Arch" : "HVMG2" },

"r3.large" : { "Arch" : "HVM64" },

"r3.xlarge" : { "Arch" : "HVM64" },

"r3.2xlarge" : { "Arch" : "HVM64" },

"r3.4xlarge" : { "Arch" : "HVM64" },

"r3.8xlarge" : { "Arch" : "HVM64" },

"i2.xlarge" : { "Arch" : "HVM64" },

"i2.2xlarge" : { "Arch" : "HVM64" },

"i2.4xlarge" : { "Arch" : "HVM64" },

"i2.8xlarge" : { "Arch" : "HVM64" },

"d2.xlarge" : { "Arch" : "HVM64" },

"d2.2xlarge" : { "Arch" : "HVM64" },

"d2.4xlarge" : { "Arch" : "HVM64" },

"d2.8xlarge" : { "Arch" : "HVM64" },

"hi1.4xlarge" : { "Arch" : "HVM64" },

"hs1.8xlarge" : { "Arch" : "HVM64" },

"cr1.8xlarge" : { "Arch" : "HVM64" },

"cc2.8xlarge" : { "Arch" : "HVM64" }

},

"AWSInstanceType2NATArch" : {

"t1.micro" : { "Arch" : "NATPV64" },

"t2.nano" : { "Arch" : "NATHVM64" },

"t2.micro" : { "Arch" : "NATHVM64" },

"t2.small" : { "Arch" : "NATHVM64" },

"t2.medium" : { "Arch" : "NATHVM64" },

"t2.large" : { "Arch" : "NATHVM64" },

"m1.small" : { "Arch" : "NATPV64" },

"m1.medium" : { "Arch" : "NATPV64" },

"m1.large" : { "Arch" : "NATPV64" },

"m1.xlarge" : { "Arch" : "NATPV64" },

"m2.xlarge" : { "Arch" : "NATPV64" },

"m2.2xlarge" : { "Arch" : "NATPV64" },

"m2.4xlarge" : { "Arch" : "NATPV64" },

"m3.medium" : { "Arch" : "NATHVM64" },

"m3.large" : { "Arch" : "NATHVM64" },

"m3.xlarge" : { "Arch" : "NATHVM64" },

"m3.2xlarge" : { "Arch" : "NATHVM64" },

"m4.large" : { "Arch" : "NATHVM64" },

"m4.xlarge" : { "Arch" : "NATHVM64" },

"m4.2xlarge" : { "Arch" : "NATHVM64" },

"m4.4xlarge" : { "Arch" : "NATHVM64" },

"m4.10xlarge" : { "Arch" : "NATHVM64" },

"c1.medium" : { "Arch" : "NATPV64" },

"c1.xlarge" : { "Arch" : "NATPV64" },

"c3.large" : { "Arch" : "NATHVM64" },

"c3.xlarge" : { "Arch" : "NATHVM64" },

"c3.2xlarge" : { "Arch" : "NATHVM64" },

"c3.4xlarge" : { "Arch" : "NATHVM64" },

"c3.8xlarge" : { "Arch" : "NATHVM64" },

"c4.large" : { "Arch" : "NATHVM64" },

"c4.xlarge" : { "Arch" : "NATHVM64" },

"c4.2xlarge" : { "Arch" : "NATHVM64" },

"c4.4xlarge" : { "Arch" : "NATHVM64" },

"c4.8xlarge" : { "Arch" : "NATHVM64" },

"g2.2xlarge" : { "Arch" : "NATHVMG2" },

"g2.8xlarge" : { "Arch" : "NATHVMG2" },

"r3.large" : { "Arch" : "NATHVM64" },

"r3.xlarge" : { "Arch" : "NATHVM64" },

"r3.2xlarge" : { "Arch" : "NATHVM64" },

"r3.4xlarge" : { "Arch" : "NATHVM64" },

"r3.8xlarge" : { "Arch" : "NATHVM64" },

"i2.xlarge" : { "Arch" : "NATHVM64" },

"i2.2xlarge" : { "Arch" : "NATHVM64" },

"i2.4xlarge" : { "Arch" : "NATHVM64" },

"i2.8xlarge" : { "Arch" : "NATHVM64" },

"d2.xlarge" : { "Arch" : "NATHVM64" },

"d2.2xlarge" : { "Arch" : "NATHVM64" },

"d2.4xlarge" : { "Arch" : "NATHVM64" },

"d2.8xlarge" : { "Arch" : "NATHVM64" },

"hi1.4xlarge" : { "Arch" : "NATHVM64" },

"hs1.8xlarge" : { "Arch" : "NATHVM64" },

"cr1.8xlarge" : { "Arch" : "NATHVM64" },

"cc2.8xlarge" : { "Arch" : "NATHVM64" }

}

,

"AWSRegionArch2AMI" : {

"us-east-1" : {"PV64" : "ami-22111148", "HVM64" : "ami-08111162", "HVMG2" : "ami-ebcec381"},

"us-west-2" : {"PV64" : "ami-792bc219", "HVM64" : "ami-c229c0a2", "HVMG2" : "ami-0f28c06f"},

"us-west-1" : {"PV64" : "ami-0e087a6e", "HVM64" : "ami-1b0f7d7b", "HVMG2" : "ami-ab9defcb"},

"eu-west-1" : {"PV64" : "ami-a5368cd6", "HVM64" : "ami-31328842", "HVMG2" : "ami-d1d652a2"},

"eu-central-1" : {"PV64" : "ami-2bde3944", "HVM64" : "ami-e2df388d", "HVMG2" : "ami-5240a73d"},

"ap-northeast-1" : {"PV64" : "ami-37020959", "HVM64" : "ami-f80e0596", "HVMG2" : "ami-34a9a35a"},

"ap-northeast-2" : {"PV64" : "NOT\_SUPPORTED", "HVM64" : "ami-6598510b", "HVMG2" : "NOT\_SUPPORTED"},

"ap-southeast-1" : {"PV64" : "ami-ff0cc79c", "HVM64" : "ami-e90dc68a", "HVMG2" : "ami-6f6ca70c"},

"ap-southeast-2" : {"PV64" : "ami-f5210196", "HVM64" : "ami-f2210191", "HVMG2" : "ami-88c1e1eb"},

"sa-east-1" : {"PV64" : "ami-661e930a", "HVM64" : "ami-1e159872", "HVMG2" : "NOT\_SUPPORTED"},

"cn-north-1" : {"PV64" : "ami-08ef2465", "HVM64" : "ami-49e22924", "HVMG2" : "NOT\_SUPPORTED"}

}

},

"Resources" : {

"WebServerInstance": {

"Type": "AWS::EC2::Instance",

"Metadata" : {

"AWS::CloudFormation::Init" : {

"configSets" : {

"InstallAndRun" : [ "Install", "Configure" ]

},

"Install" : {

"packages" : {

"yum" : {

"mysql" : [],

"mysql-server" : [],

"mysql-libs" : [],

"httpd" : [],

"php" : [],

"php-mysql" : []

}

},

"files" : {

"/var/www/html/index.php" : {

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

"<html>\n",

" <head>\n",

" <title>AWS CloudFormation PHP Sample</title>\n",

" <meta http-equiv=\"Content-Type\" content=\"text/html; charset=ISO-8859-1\">\n",

" </head>\n",

" <body>\n",

" <h1>Welcome to the AWS CloudFormation PHP Sample</h1>\n",

" <p/>\n",

" <?php\n",

" // Print out the current data and time\n",

" print \"The Current Date and Time is: <br/>\";\n",

" print date(\"g:i A l, F j Y.\");\n",

" ?>\n",

" <p/>\n",

" <?php\n",

" // Setup a handle for CURL\n",

" $curl\_handle=curl\_init();\n",

" curl\_setopt($curl\_handle,CURLOPT\_CONNECTTIMEOUT,2);\n",

" curl\_setopt($curl\_handle,CURLOPT\_RETURNTRANSFER,1);\n",

" // Get the hostname of the intance from the instance metadata\n",

" curl\_setopt($curl\_handle,CURLOPT\_URL,'http://169.254.169.254/latest/meta-data/public-hostname');\n",

" $hostname = curl\_exec($curl\_handle);\n",

" if (empty($hostname))\n",

" {\n",

" print \"Sorry, for some reason, we got no hostname back <br />\";\n",

" }\n",

" else\n",

" {\n",

" print \"Server = \" . $hostname . \"<br />\";\n",

" }\n",

" // Get the instance-id of the intance from the instance metadata\n",

" curl\_setopt($curl\_handle,CURLOPT\_URL,'http://169.254.169.254/latest/meta-data/instance-id');\n",

" $instanceid = curl\_exec($curl\_handle);\n",

" if (empty($instanceid))\n",

" {\n",

" print \"Sorry, for some reason, we got no instance id back <br />\";\n",

" }\n",

" else\n",

" {\n",

" print \"EC2 instance-id = \" . $instanceid . \"<br />\";\n",

" }\n",

" $Database = \"localhost\";\n",

" $DBUser = \"", {"Ref" : "DBUser"}, "\";\n",

" $DBPassword = \"", {"Ref" : "DBPassword"}, "\";\n",

" print \"Database = \" . $Database . \"<br />\";\n",

" $dbconnection = mysql\_connect($Database, $DBUser, $DBPassword)\n",

" or die(\"Could not connect: \" . mysql\_error());\n",

" print (\"Connected to $Database successfully\");\n",

" mysql\_close($dbconnection);\n",

" ?>\n",

" <h2>PHP Information</h2>\n",

" <p/>\n",

" <?php\n",

" phpinfo();\n",

" ?>\n",

" </body>\n",

"</html>\n"

]]},

"mode" : "000600",

"owner" : "apache",

"group" : "apache"

},

"/tmp/setup.mysql" : {

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

"CREATE DATABASE ", { "Ref" : "DBName" }, ";\n",

"GRANT ALL ON ", { "Ref" : "DBName" }, ".\* TO '", { "Ref" : "DBUser" }, "'@localhost IDENTIFIED BY '", { "Ref" : "DBPassword" }, "';\n"

]]},

"mode" : "000400",

"owner" : "root",

"group" : "root"

},

"/etc/cfn/cfn-hup.conf" : {

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

"[main]\n",

"stack=", { "Ref" : "AWS::StackId" }, "\n",

"region=", { "Ref" : "AWS::Region" }, "\n"

]]},

"mode" : "000400",

"owner" : "root",

"group" : "root"

},

"/etc/cfn/hooks.d/cfn-auto-reloader.conf" : {

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

"[cfn-auto-reloader-hook]\n",

"triggers=post.update\n",

"path=Resources.WebServerInstance.Metadata.AWS::CloudFormation::Init\n",

"action=/opt/aws/bin/cfn-init -v ",

" --stack ", { "Ref" : "AWS::StackName" },

" --resource WebServerInstance ",

" --configsets InstallAndRun ",

" --region ", { "Ref" : "AWS::Region" }, "\n",

"runas=root\n"

]]}

}

},

"services" : {

"sysvinit" : {

"mysqld" : { "enabled" : "true", "ensureRunning" : "true" },

"httpd" : { "enabled" : "true", "ensureRunning" : "true" },

"cfn-hup" : { "enabled" : "true", "ensureRunning" : "true",

"files" : ["/etc/cfn/cfn-hup.conf", "/etc/cfn/hooks.d/cfn-auto-reloader.conf"]}

}

}

},

"Configure" : {

"commands" : {

"01\_set\_mysql\_root\_password" : {

"command" : { "Fn::Join" : ["", ["mysqladmin -u root password '", { "Ref" : "DBRootPassword" }, "'"]]},

"test" : { "Fn::Join" : ["", ["$(mysql ", { "Ref" : "DBName" }, " -u root --password='", { "Ref" : "DBRootPassword" }, "' >/dev/null 2>&1 </dev/null); (( $? != 0 ))"]]}

},

"02\_create\_database" : {

"command" : { "Fn::Join" : ["", ["mysql -u root --password='", { "Ref" : "DBRootPassword" }, "' < /tmp/setup.mysql"]]},

"test" : { "Fn::Join" : ["", ["$(mysql ", { "Ref" : "DBName" }, " -u root --password='", { "Ref" : "DBRootPassword" }, "' >/dev/null 2>&1 </dev/null); (( $? != 0 ))"]]}

}

}

}

}

},

"Properties": {

"ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region" },

{ "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "InstanceType" }, "Arch" ] } ] },

"InstanceType" : { "Ref" : "InstanceType" },

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

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

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

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

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

"# Install the files and packages from the metadata\n",

"/opt/aws/bin/cfn-init -v ",

" --stack ", { "Ref" : "AWS::StackName" },

" --resource WebServerInstance ",

" --configsets InstallAndRun ",

" --region ", { "Ref" : "AWS::Region" }, "\n",

"# Signal the status from cfn-init\n",

"/opt/aws/bin/cfn-signal -e $? ",

" --stack ", { "Ref" : "AWS::StackName" },

" --resource WebServerInstance ",

" --region ", { "Ref" : "AWS::Region" }, "\n"

]]}}

},

"CreationPolicy" : {

"ResourceSignal" : {

"Timeout" : "PT5M"

}

}

},

"WebServerSecurityGroup" : {

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

"Properties" : {

"GroupDescription" : "Enable HTTP access via port 80",

"SecurityGroupIngress" : [

{"IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "CidrIp" : "0.0.0.0/0"},

{"IpProtocol" : "tcp", "FromPort" : "22", "ToPort" : "22", "CidrIp" : { "Ref" : "SSHLocation"}}

]

}

}

},

"Outputs" : {

"WebsiteURL" : {

"Description" : "URL for newly created LAMP stack",

"Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" : [ "WebServerInstance", "PublicDnsName" ]}]] }

}

}

}

**SUBMISSION PROCESS**

**Note: Once you complete the assignment, please upload all files/docs to a Git repo and share the link in reply to this email.**