**Deploy a website in AWS.**

We can do it in 2 ways.

1.using AWS amplify

2. In the aws management console search aws amplify

3.click on deliver.

4.select deploy without git provider and click continue

5, Now drag your website into it.

6.click and save.

7. when deployment is completed click on the link to see live.

**Another way.**

1.Create a website.

2.Now zip it and save it into a s3 bucket and make it public.

3. spawn an ec2 instance.

4.Now take the ec2 instance public ip address and login using putty and the saved key to login.

5. yum install update

6. Yum install httpd -y ( apache webserver)

7.now type wget and the public url for s3 and download it.

8.Now move the files to var/www/html

9.now take the ec2 instance public ip address and paste it in a browser it will display the contents.

**Deploying using a docker.**

1.Download a website.

2.wget and the public url.

3.Now create a docker file.

4.vi dockerfile

Inside the write this code.

From httpd

Copy . /usr/local/apache/htdocs

:wq!

5.docker build -t website .

6.docker run -itd( for running in background) -p 80:80 -name website website

7.Now take a docker public ip on a docker host and put it in the web browser.

**Inject an environment variable (SECRET\_WORD) in the Docker container. The value of SECRET\_WORD should be the secret word discovered on the index page of the application.**

1.docker run -it -e SECRET\_WORD=”secret” ubuntu bash

Env parameter is used if we want to write a file with lots of variables.

**Deploy a load balancer in front of the app**

|  |
| --- |
| http { |
|  |  |
|  | upstream all { |
|  | server localhost:2222; |
|  | server localhost:3333; |
|  | } |
|  |  |
|  | server { |
|  | listen 8080; |
|  | location / { |
|  | proxy\_pass http://all/; |
|  | } |
|  | } |
|  |  |
|  | } |
|  |  |
|  |  |
|  | events { } |

2.vi docker-file

FROM ngnix

COPY nigix.conf /etc/ngnix/ngnix.conf

:wq!

2.docker build -t ngnix

3.docker run -p 8080:8080 ngnixapp

|  |
| --- |
| AWSTemplateFormatVersion: '2010-09-09' |
|  | Metadata: |
|  | License: Apache-2.0 |
|  | Description: 'AWS CloudFormation Sample Template EC2InstanceWithSecurityGroupSample: |
|  | Create an Amazon EC2 instance running the Amazon Linux AMI. The AMI is chosen based |
|  | on the region in which the stack is run. This example creates an EC2 security group |
|  | for the instance to give you SSH access. \*\*WARNING\*\* This template creates an Amazon |
|  | EC2 instance. You will be billed for the AWS resources used if you create a stack |
|  | from this 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. |
|  | InstanceType: |
|  | Description: WebServer EC2 instance type |
|  | Type: String |
|  | Default: t3.small |
|  | AllowedValues: [t2.nano, t2.micro, t2.small, t2.medium, t2.large, t2.xlarge, t2.2xlarge, |
|  | t3.nano, t3.micro, t3.small, t3.medium, t3.large, t3.xlarge, t3.2xlarge, |
|  | m4.large, m4.xlarge, m4.2xlarge, m4.4xlarge, m4.10xlarge, |
|  | m5.large, m5.xlarge, m5.2xlarge, m5.4xlarge, |
|  | c5.large, c5.xlarge, c5.2xlarge, c5.4xlarge, c5.9xlarge, |
|  | g3.8xlarge, |
|  | r5.large, r5.xlarge, r5.2xlarge, r5.4xlarge, r3.12xlarge, |
|  | i3.xlarge, i3.2xlarge, i3.4xlarge, i3.8xlarge, |
|  | d2.xlarge, d2.2xlarge, d2.4xlarge, d2.8xlarge] |
|  | 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. |
|  | LatestAmiId: |
|  | Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>' |
|  | Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86\_64-gp2' |
|  | Resources: |
|  | EC2Instance: |
|  | Type: AWS::EC2::Instance |
|  | Properties: |
|  | InstanceType: !Ref 'InstanceType' |
|  | SecurityGroups: [!Ref 'InstanceSecurityGroup'] |
|  | KeyName: !Ref 'KeyName' |
|  | ImageId: !Ref 'LatestAmiId' |
|  | InstanceSecurityGroup: |
|  | Type: AWS::EC2::SecurityGroup |
|  | Properties: |
|  | GroupDescription: Enable SSH access via port 22 |
|  | SecurityGroupIngress: |
|  | - IpProtocol: tcp |
|  | FromPort: 22 |
|  | ToPort: 22 |
|  | CidrIp: !Ref 'SSHLocation' |
|  | Outputs: |
|  | InstanceId: |
|  | Description: InstanceId of the newly created EC2 instance |
|  | Value: !Ref 'EC2Instance' |
|  | AZ: |
|  | Description: Availability Zone of the newly created EC2 instance |
|  | Value: !GetAtt [EC2Instance, AvailabilityZone] |
|  | PublicDNS: |
|  | Description: Public DNSName of the newly created EC2 instance |
|  | Value: !GetAtt [EC2Instance, PublicDnsName] |
|  | PublicIP: |
|  | Description: Public IP address of the newly created EC2 instance |
|  | Value: !GetAtt [EC2Instance, PublicIp] |

TLS:-

We can do SSL/TLS in 3 different ways. TLS, mTLS, TLS with kerboros ( where ative directory is involved.IHave done all the 3 of them.