Deploy Svelte Website on EC2 using Terraform

- We will provision the EC2 for Static & Dynamic Web Hosting.
- We will deploy it in a custom Virtual Private Cloud for isolation.
- We will configure the server with Nginx as a Reverse Proxy and attach an SSL Certificate for secure web access.
- We will also create a Route53 Hosted Zone and add a Route53 Record to access the website using the domain name.
- We will create all these resources using Terraform as an Infrastructure as Code.

Prerequisites

---

1. AWS Account with IAM User Access Keys
2. Terraform installed
3. Website repository
4. Domain name

Generate SSH Key Pair

First, we will generate an **SSH Key Pair** to SSH into the EC2 server once it is created and start the Svelte with Nodejs.

Steps

1. Open the Powershell Window.
2. Run the following command to generate the ssh key pair and enter the location where you want to save the ssh key (C:/Users/user/.ssh/aws/ssh-keys/):

```
ssh-keygen -t ed25519
```

3. SSH Key will be generated.

Write Terraform Configuration files

Now we will write Terraform configuration files for AWS resources using predefined modules available on the internet.

Steps

1. Create the **ec2-website-terraform** project directory.

2. The folder structure for the above-created directory is as follows:

```
ec2-website-terraform
├──.terraform.lock.hcl
├──locals.tf
├──main.tf
├──outputs.tf
├──providers.tf
├──terraform.tfstate
├──terraform.tfstate.backup
└──.terraform
```

> We need to only create *providers.tf*, *main.tf*, *outputs.tf*, & *locals.tf* file. Other files are generated while initiating terraform.

3. Create a *providers.tf* file inside the above-created directory.
4. Inside it, define the following:
    - terraform
        - required_providers
    - provider
        - aws
5. Click code for reference.
6. The definition of *providers.tf* file is complete.
7. Now create the *main.tf* file.
8. Define the following modules inside it:
    - module.vpc
    - module. ec2
9. Click code for reference.
10. The definition of *main.tf* file is complete.
11. Now we will create *outputs.tf* file.
12. Inside it, define the following outputs
    - output.ec2-instance-public-ip
13. Click code for reference.
14. The definition of *outputs.tf* file is complete.
15. Now we will create *locals.tf* file.
16. Inside it, define the following variables:
    - local.aws-region
    - local.vpc-properties
    - local.ec2-properties
17. Click code for reference.
18. The definition of *locals.tf* file is complete.

> Ensure you give the appropriate values to the variables defined in *locals.tf* file.
> In *ec2-properties.ec2-instance-public-key*, provide the above generated public SSH key path.

Provisioning the Infrastructure

Now we will provision the AWS infrastructure by applying the above-created Terraform configuration files.

> Ensure AWS CLI is configured with appropriate IAM User Access Keys with enough permissions.

Steps:

1. Open the PowerShell Window.
2. Change the directory to the above-created ec2-website-terraform directory using the `cd` command.
3. Run the `terraform fmt -recursive` command to format the syntax of the files.
4. Run the `terraform init` command to initialize the terraform.
5. Run the `terraform validate` command to validate the configuration files.
6. Run the `terraform plan` command to plan the resources to be created.
7. Run the `terraform apply` command and if prompted, type `yes` to provision the infrastructure.
8. Run the `terraform output` command to get the values of defined variables in outputs.tf file.
9. Head to the **AWS Console**, and verify the created resources.

Route53 Configuration

Now we will configure the Route53 service for domain routing to the website.

Steps

1. Login to the AWS console and search for the **Route-53** service.
2. Click open the Route-53 console.
3. In the left plane of the window, click on `Hosted zones`.
4. Create a new hosted zone for your domain name e.g. `example.com`.
5. On completing, two records of type `NS` and `SOA` gets created here.
6. Copy all four values from column `Value/Route traffic to` of `NS` record.
7. Go to your domain provider's website and add these copied nameservers in place of your domain's original nameservers. This will dedicate your domain to AWS.
8. Again go to the hosted zone added in the AWS Route-53 console.
9. Add a `dev.example.com`(replace with your domain name) record of type `A` pointing to your EC2 instance's IP address received from running `terraform output` command.

SSH Into EC2 Server

Now we will SSH into the EC2 instance and configure the server for website deployment.

Steps

1. Open the Powershell Window.
2. Run the following command to SSH into EC2 server and substitute the <*private-ssh-key-path*> with the above generated private ssh key path and <*ec2-instance-public-ip*> with the server IP received from `terraform output` command:

```
    ssh -o StrictHostKeyChecking=no -i <*private-ssh-key-path*> ec2-user@<*ec2-
instance-public-ip*>
```

3. Once you enter the server, run the following commands to install the necessary dependencies for deployment:

```
    sudo yum update
    sudo yum install -y nodejs npm
    sudo npm install -g pm2
    sudo yum install -y nginx
    sudo yum install -y git
    sudo yum install -y certbot python3-certbot-nginx
    sudo npm install @sveltejs/adapter-node
```

4. Now clone the svelte git repository that you want to deploy by running and substituting the following link with your desired git repository link:

```
    git clone <https://github.com/account-name/repository-name.git>
```

5. Navigate to the website folder using the `cd` command and run the following commands to start the Nodejs server:

```
    sudo npm install @sveltejs/adapter-node
    sudo npm install
    sudo npm run build
    sudo pm2 start build/index.js --name "sveltekit-app"
    sudo pm2 save
    sudo pm2 startup
```

6. Now we will configure the Nginx as a Reverse Proxy with HTTPS to secure the web connection.
7. Click code and copy the default.conf file and paste it into the vim terminal opened by running the following command. Also, replace the <*dev.example.com*> with your desired domain name. Click `esc` and type `:wq` to write the file.
8. Run the following commands to configure the Nginx:

```
    sudo vim /etc/nginx/conf.d/default.conf
    sudo systemctl restart nginx
    sudo certbot --nginx -d dev.example.com
    sudo certbot renew --dry-run
```

9. Nginx configuration is complete. Try accessing the website on the browser.

Destroy the provisioned infrastructure

Lastly, we will destroy the above-created resources.

Steps

1. To destroy infrastructure, open the Powershell Window and change the directory to the above-created **ec2-website-terraform** directory using the `cd` command.
2. Run `terraform destroy` & if prompted, type `yes`.
3. Infrastructure will be destroyed.