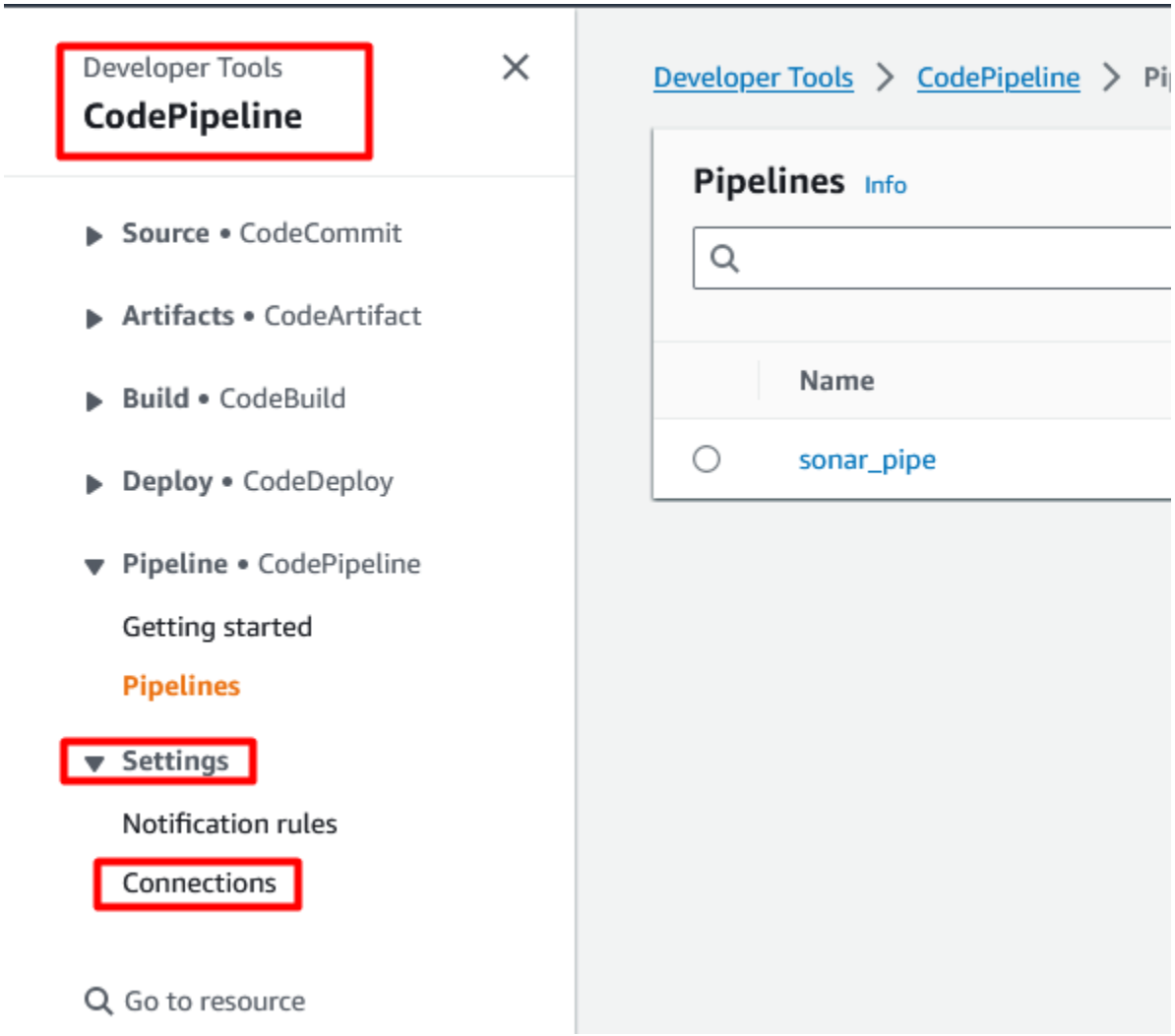
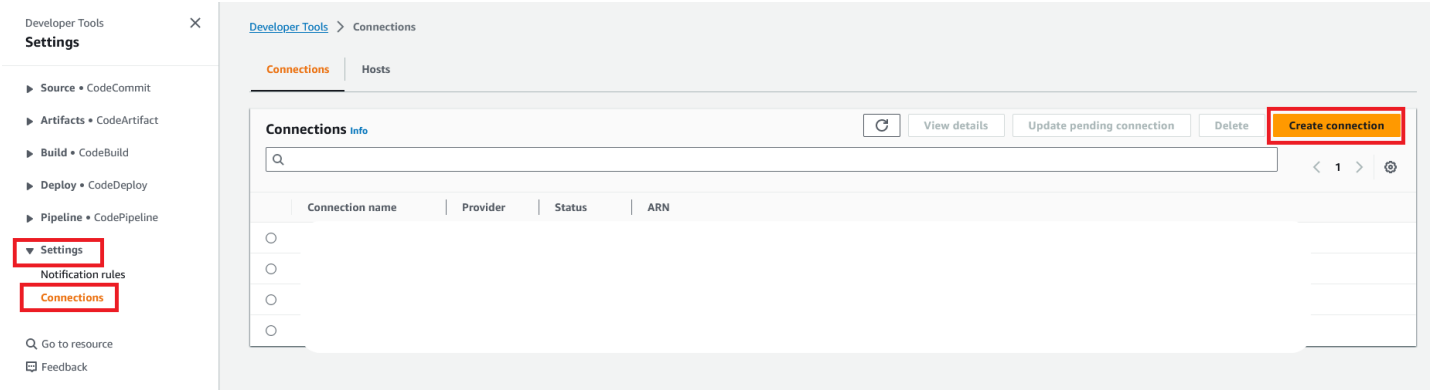
1. Create Code Star connection in AWS

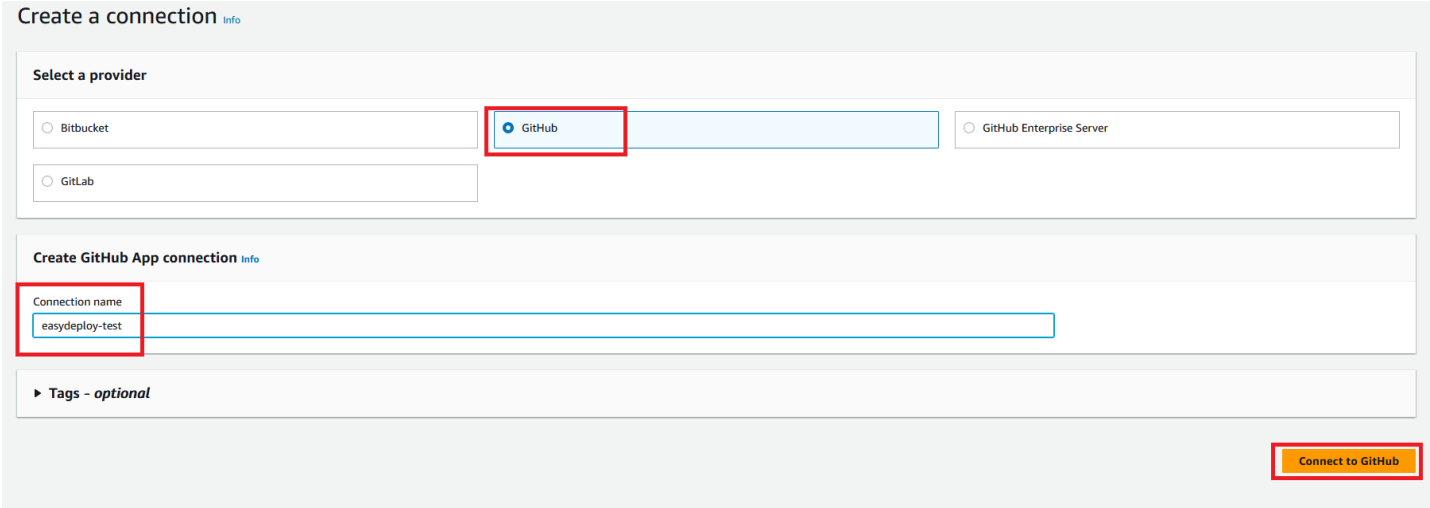
Open your AWS console **All services**→ **CodePipeline** page.
On the left navigation panel click **Settings** → **Connections**



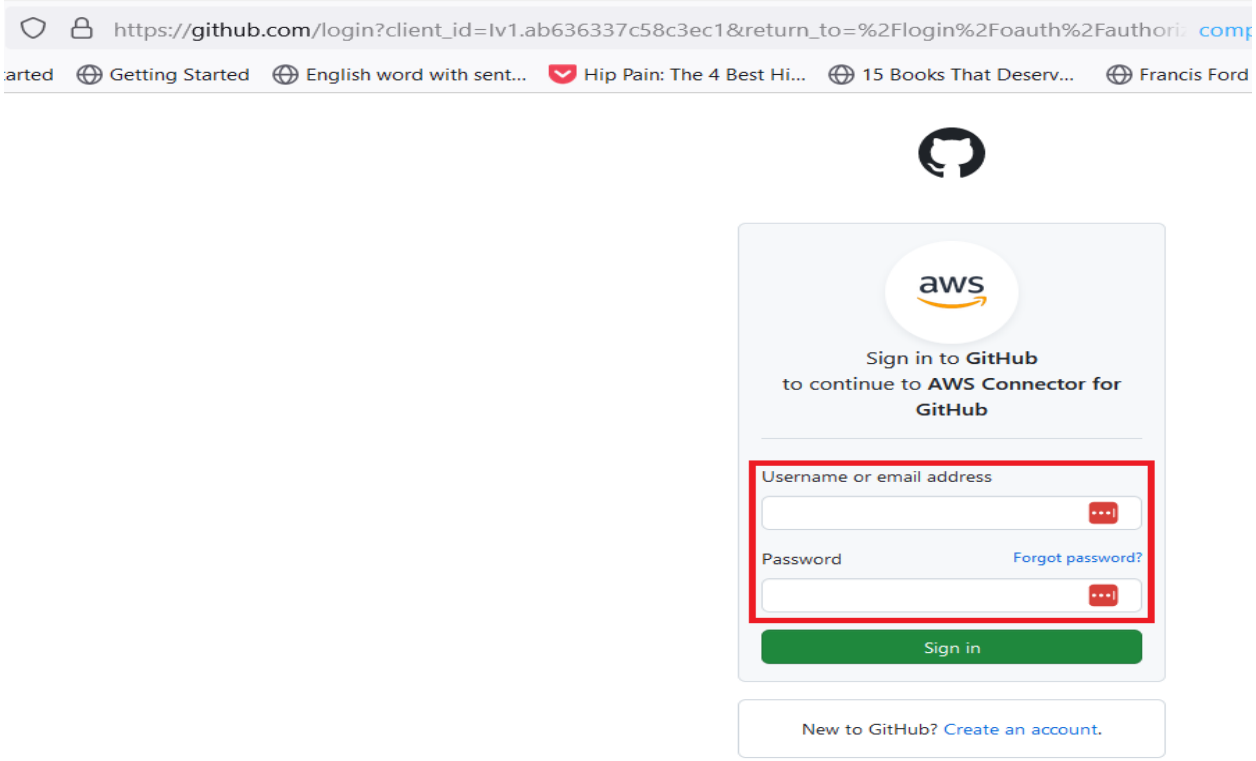
Now Click on **Create Connection**



- **Select a provider→ Github(**as your source code is in github repo**)** for provider
- **Connection name**→ easydeploy-test (any name of your choice),

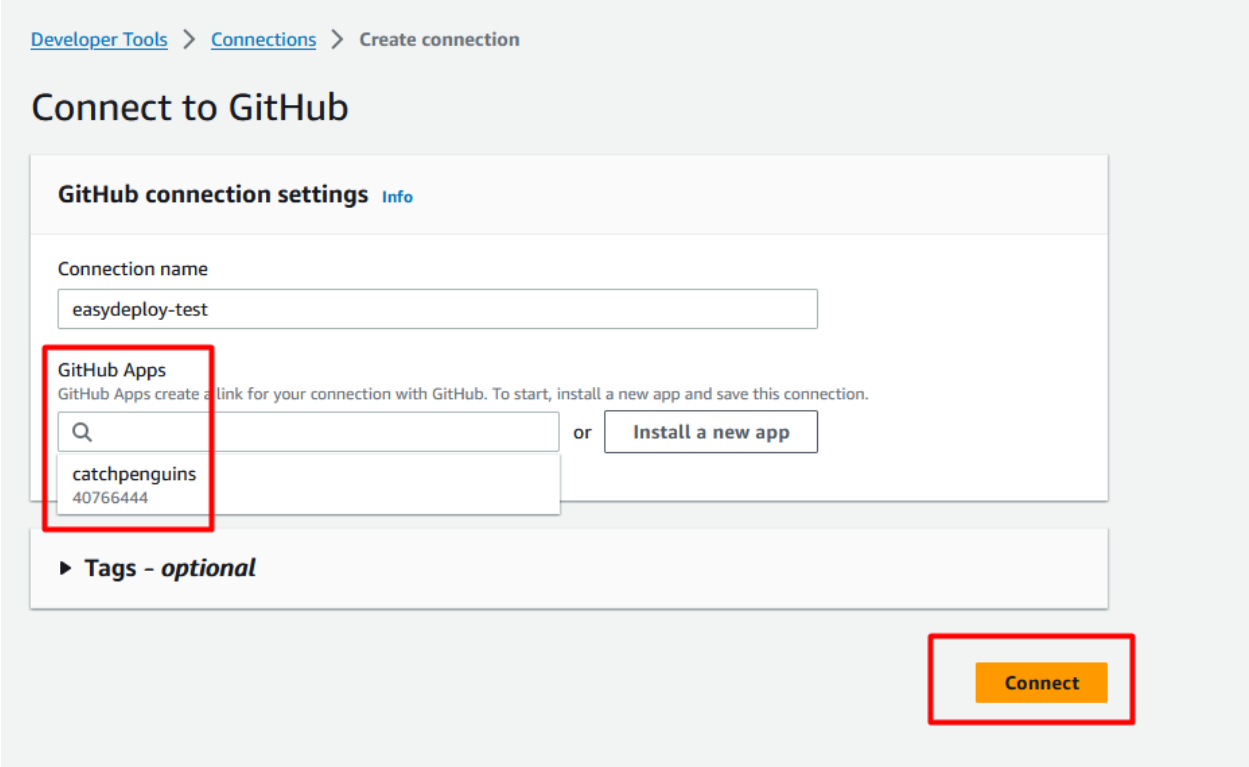


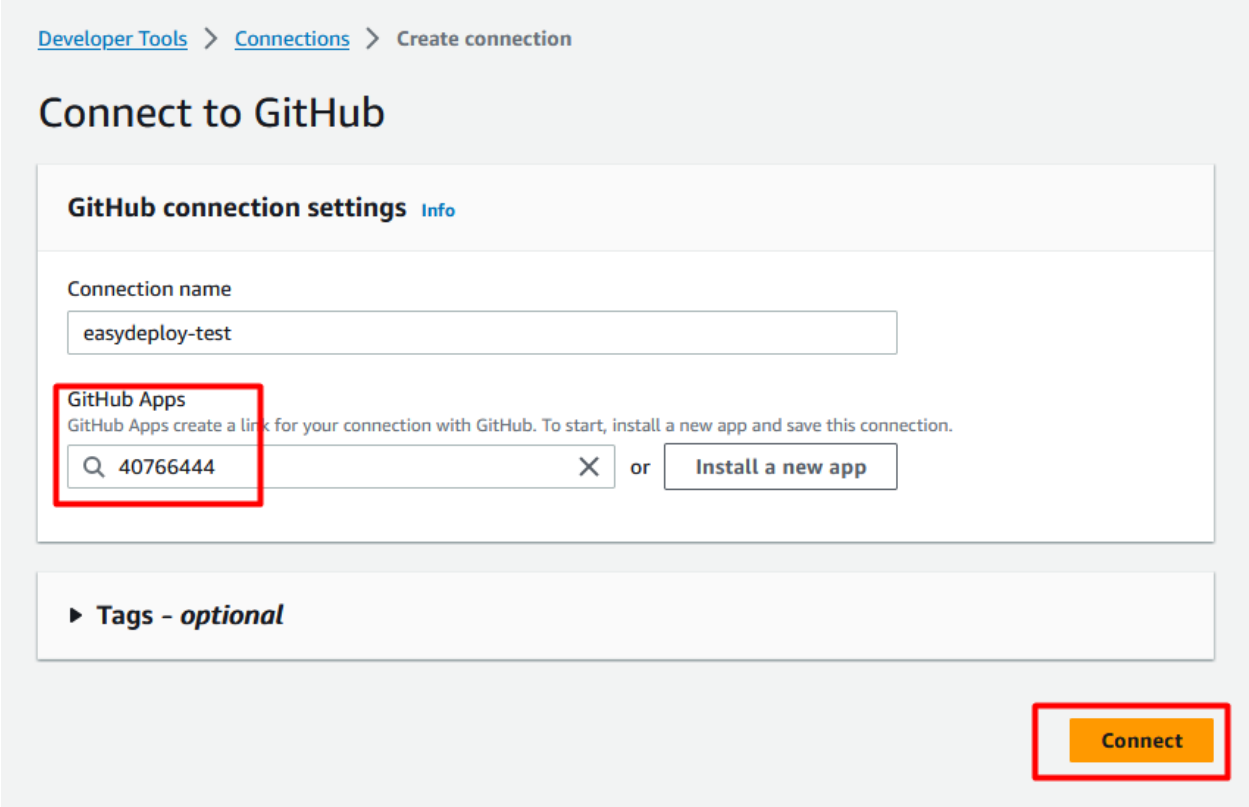
then click **Connect to Github (**it will open the Credentials tabs in your browser kindly authenticate it**)**



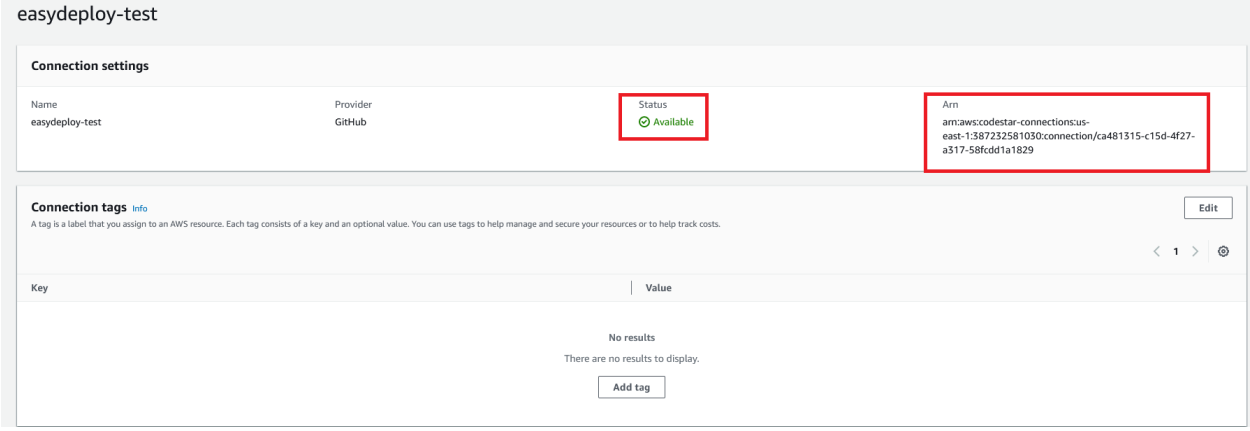
Once the authentication is finished, it will directly show the below page in AWS account in this page

Click on **GitHub Apps**→and select the connection from the Drop down(after selecting it will show the random number) and click **Connect**.





Now you can see the Connection status is in the **Available** state.
Copy the ARN marked in the screenshot below and note it somewhere else.
This Connection ARN needs to be used in the CodePipeline terraform script



Procedure to Execute Terraform Script

Step 1 Install terraform

Use the below link and Choose your appropriate OS and install the terraform

https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

Step 2 Copy the terraform script Zip file

Unzip the terraform script Zip file(you will find many files with **.tf** extension) and then enter into the folder and execute the below command

```
Terraform init
```

This is a one-time run command you don't need to execute this command each time you run terraform.

```
PS D:\AWS\CUSTOMERS\New-client> terraform init

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.12.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Then run the below command

```
Terraform plan
```

The above command will give the resources which its going to provision in order for this to work you need it will prompt to input the below details like in the below screenshot. You have to enter the details every time you run the command.

These are the details you have to provide while executing the terraform script.

1. **Access_key** - AWS Access Key ID of IAM User
2. **Secret_key** - AWS Secret Access Key ID of IAM User
3. **App_name** - Common name for this Infrastructure services
4. **Branch_name** - Name of the source code branch
5. **Codestar_connection_arn** - Code Star connection ARN
6. **Region** - Region for AWS
7. **Repository_id** - ID of the GitHub repo (username/repository)

```
PS D:\AWS\CUSTOMERS\isebarn-terraform terraform plan
var.access_key
  Enter the AWS Access Key ID

  Enter a value: AKIAVUKG4IWTF2MOIENS

var.app_name
  Name for ecs service, task, target-group, Load balancer

  Enter a value: easydeploy

var.branch_name
  Name of the source code branch name

  Enter a value: master

var.codestar_connection_arn
  ARN of the code star connection

  Enter a value: arn:aws:codestar-connections:us-east-1:387232581030:connection/cdabfcbe-3f2e-4f0b-8f5e-d05fe179ad57

var.region
  Enter the region for your infrastructure

  Enter a value: us-east-1

var.repository_id
  Repository ID of Github

  Enter a value: isebarn/test_nuxt

var.secret_key
  Enter the AWS Secret Access Key

  Enter a value: fw6WLF1Ng54s6nX59bD9egz9/FWnRK2L7I3FaV8+
```

Once the plan command runs, it will show you the number of resources it is going to provision and also it will show if there is any error in the script

```
  # module.vpc.aws_vpc.this[0] will be created
  + resource "aws_vpc" "this" {
      + arn                                  = (known after apply)
      + assign_generated_ipv6_cidr_block     = false
      + cidr_block                           = "10.10.0.0/16"
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_dns_hostnames                 = true
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = false
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Env"  = "production"
          + "Name" = "easydeploy"
        }
      + tags_all                             = {
          + "Env"  = "production"
          + "Name" = "easydeploy"
        }
    }

Plan: 42 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + dns_name = (known after apply)
```

Now finally in order to provision the resources, run the following command to execute the provision of the required resources

```
terraform apply
```

Once you run the above command **input all the parameter details** that prompt.

```
PS D:\AWS\CUSTOMERS\New-client> terraform apply
var.access_key
  Enter the AWS Access Key ID

  Enter a value:
```

Confirm the creation by entering **yes** to create resources.

```
      + tags                                   = {
          + "Env"  = "production"
          + "Name" = "easydeploy"
        }
      + tags_all                               = {
          + "Env"  = "production"
          + "Name" = "easydeploy"
        }
    }

Plan: 42 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + dns_name = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

It starts to create the resources on your AWS account.

```
  Enter a value: yes

module.pipeline.aws_iam_role.build_role[0]: Creating...
module.pipeline.aws_iam_role.pipeline_role: Creating...
module.security-group-lb.aws_security_group.this[0]: Creating...
module.security-group-ecs.aws_security_group.this[0]: Creating...
module.load-balancing.aws_lb_target_group.this[0]: Creating...
module.pipeline.aws_s3_bucket.this[0]: Creating...
module.pipeline.aws_iam_role.pipeline_role: Creation complete after 2s [id=easydeploy-pipeline-role]
```

The completion of the resource creation would take 5 minutes.
Once the Provision is completed it shows **Apply complete!** like the below picture.
You can see an output which is the **endpoint of the Load balancer.**

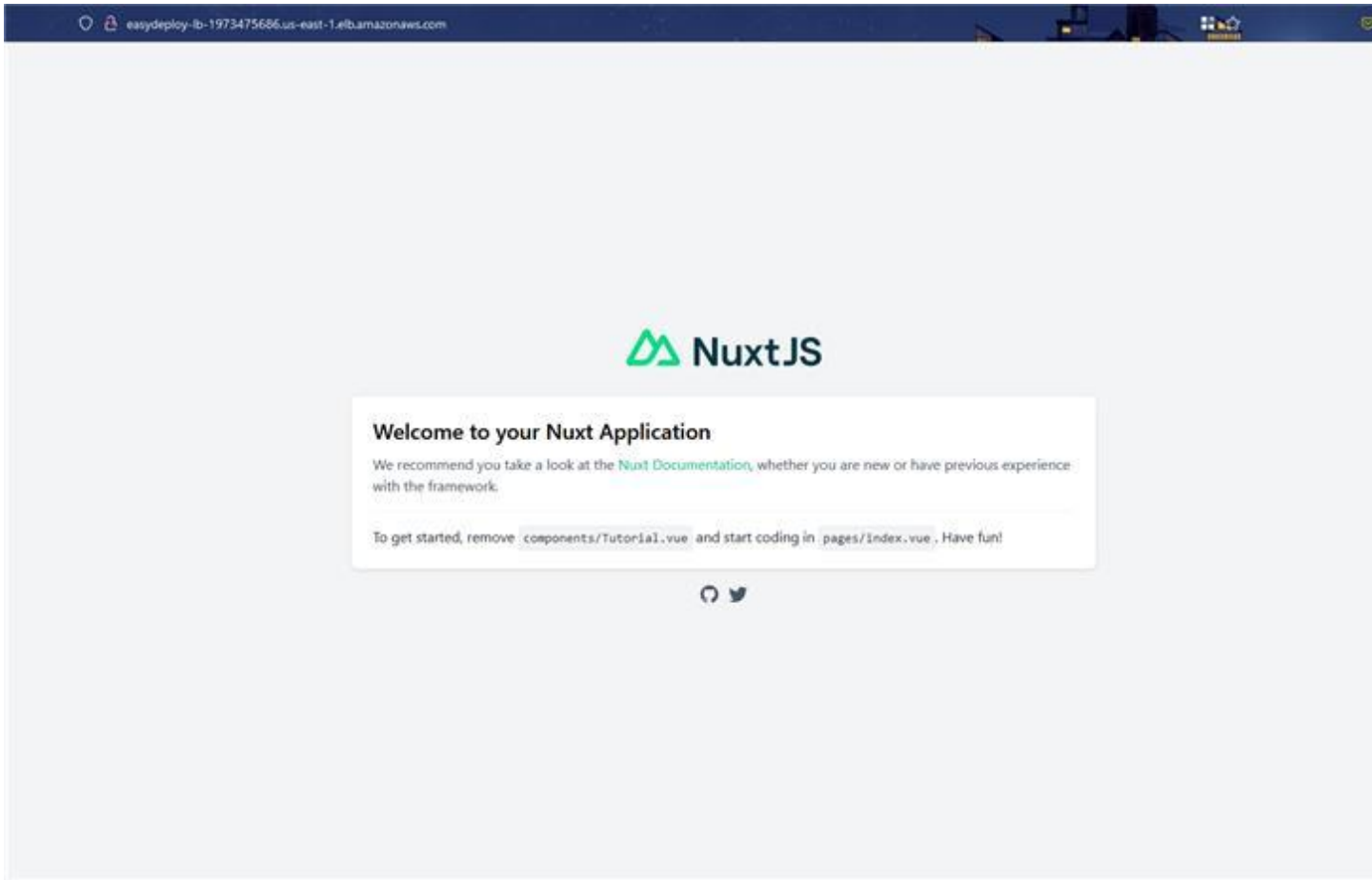
```
module.pipeline.aws_codebuild_project.this[0]: Creation complete after 2s [id=arn:aws:codebuild:us-east-1:387232581030:project/easydep
module.ecs.aws_ecs_cluster.this[0]: Still creating... [10s elapsed]
module.ecs.aws_ecs_cluster.this[0]: Creation complete after 13s [id=arn:aws:ecs:us-east-1:387232581030:cluster/easydeploy]
module.ecs.aws_ecs_service.this[0]: Creating...
module.ecs.aws_ecs_service.this[0]: Creation complete after 1s [id=arn:aws:ecs:us-east-1:387232581030:service/easydeploy/easydeploy]
module.ecs.aws_appautoscaling_target.this[0]: Creating...
module.pipeline.aws_codepipeline.this_2[0]: Creating...
module.ecs.aws_appautoscaling_target.this[0]: Creation complete after 2s [id=service/easydeploy/easydeploy]
module.ecs.aws_appautoscaling_policy.this[0]: Creating...
module.pipeline.aws_codepipeline.this_2[0]: Creation complete after 2s [id=easydeploy]
module.ecs.aws_appautoscaling_policy.this[0]: Creation complete after 1s [id=easydeploy-CpuUtilization]

Apply complete! Resources: 42 added, 0 changed, 0 destroyed.

Outputs:

dns_name = "easydeploy-LB-1973475686.us-east-1.elb.amazonaws.com"
```

Once the Apply is completed wait **10 more minutes** to server comes up
After 10 minutes, you can access your application from the browser using the **dns_name** from the outputs.



NOTE:

Once the terraform apply is completed you can see a new file created (**terraform.tfstate**). This file is important for future reference or any more changes so do not delete it.