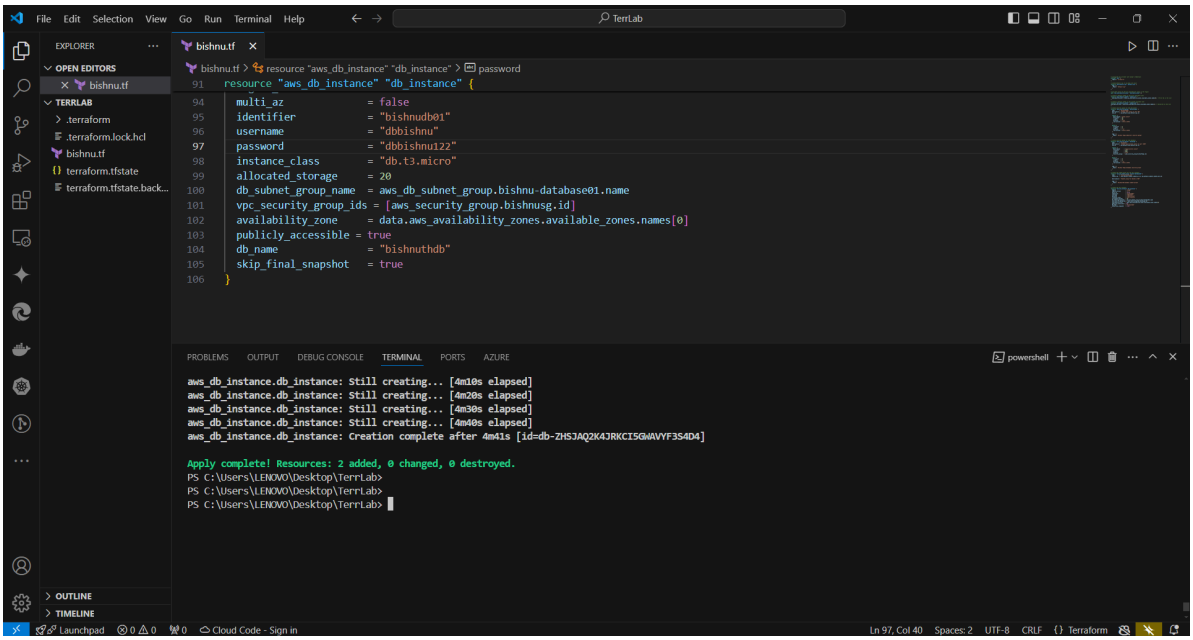
RDS Lab POC

Provisioning RDS using MySQL database:

Steps: Download and set up vs-code, terraform latest, gitbash, and aws cli tools that are necessary to perform this lab locally.

After setup done, write tf file provided in my git repo and snapshot looks like following:

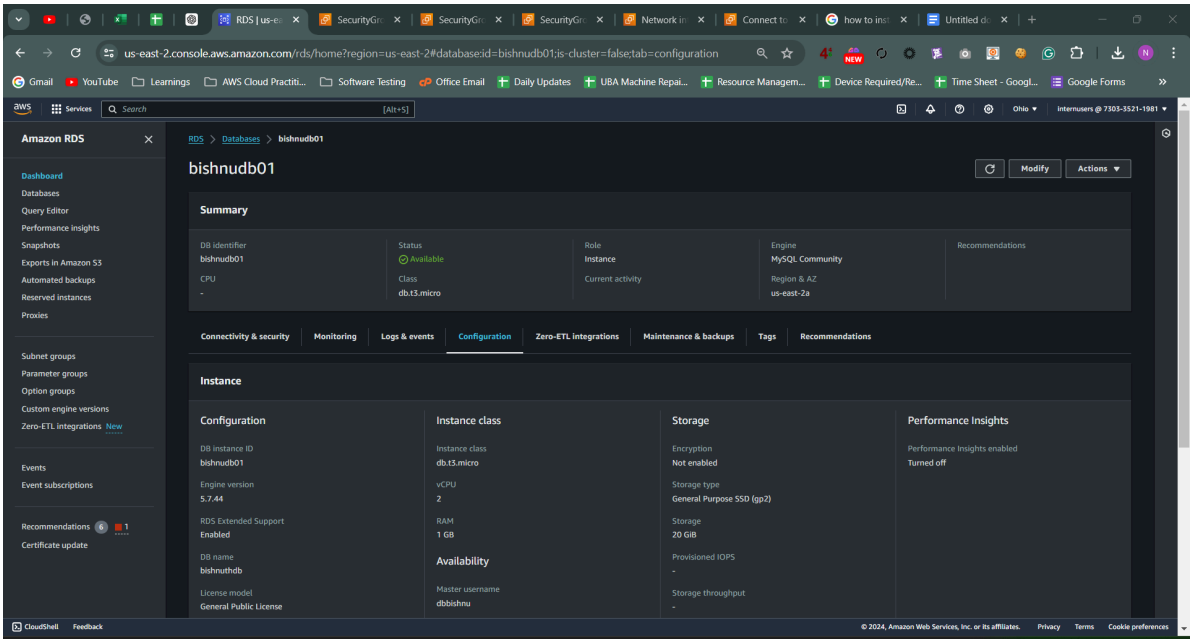


Coding parts are done, and terraform commands are executed in cli.
Commands to execute terraform files:

1. terraform init this command use for initialized terraform in respected folder.
2. terraform plan this command is use for dry-run check for output after apply the terraform apply command.
3. terraform validate this command validate our code is written and valid or not.
4. terraform apply and terraform apply -auto-approve both command use for apply the changes to provision accordingly, -auto-approve use for automatically approve without manually entering yes parameter.
5. terraform destroy command use for destroy the provisioned infrastructure accordingly.

After using these commands, our terraform code will provision the resources and also can destroy the resource by one click.

For eg. we did provision RDS database in ec2 using terraform code:



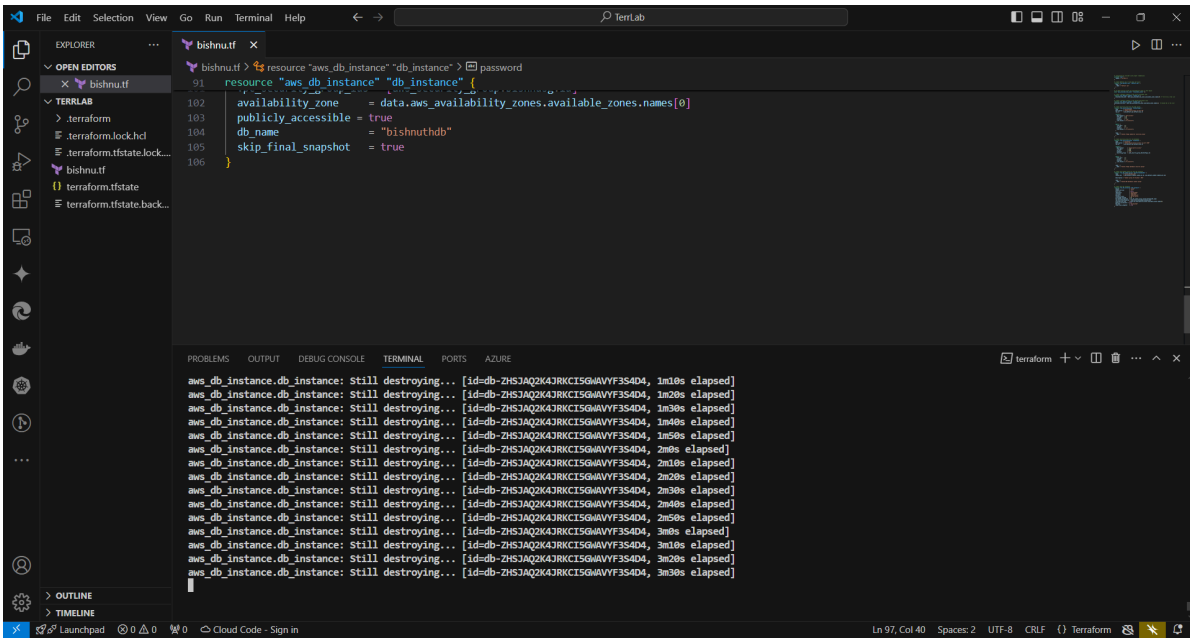
After making db, we need to tweak our security group to access the 3306 port by modifying the security group in security groups sections.

Let's access our db accordingly. I am using windows so need another vm running with linux, so used ec2 instance for db connection check:

```
root@ip-172-31-17-10:/home/ubuntu# mysql -h bishnudb01.c780ygmcy1iz.us-east-2.rds.amazonaws.com -u dbbishnu -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.7.44 Please upgrade to 8.0 or opt-in to the paid RDS Extended Support service before 5.7 reac
hes end of standard support on 29 February, 2024: https://a.co/hQqiIn0

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| bishnuthdb         |
| innodb             |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.00 sec)
```

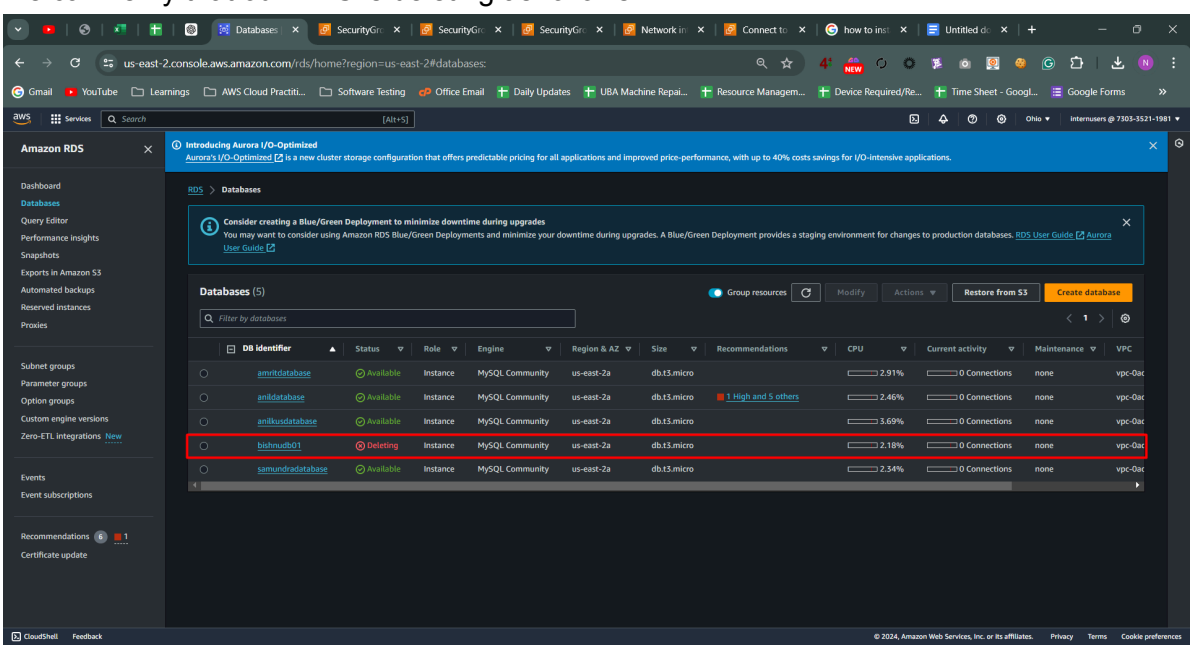
DB is accessible and working fine now.

Destroying: After created the resources and we did check is it working or not, we se our db is perfectly working fine. So let's destroy it: using terraform destroy command.

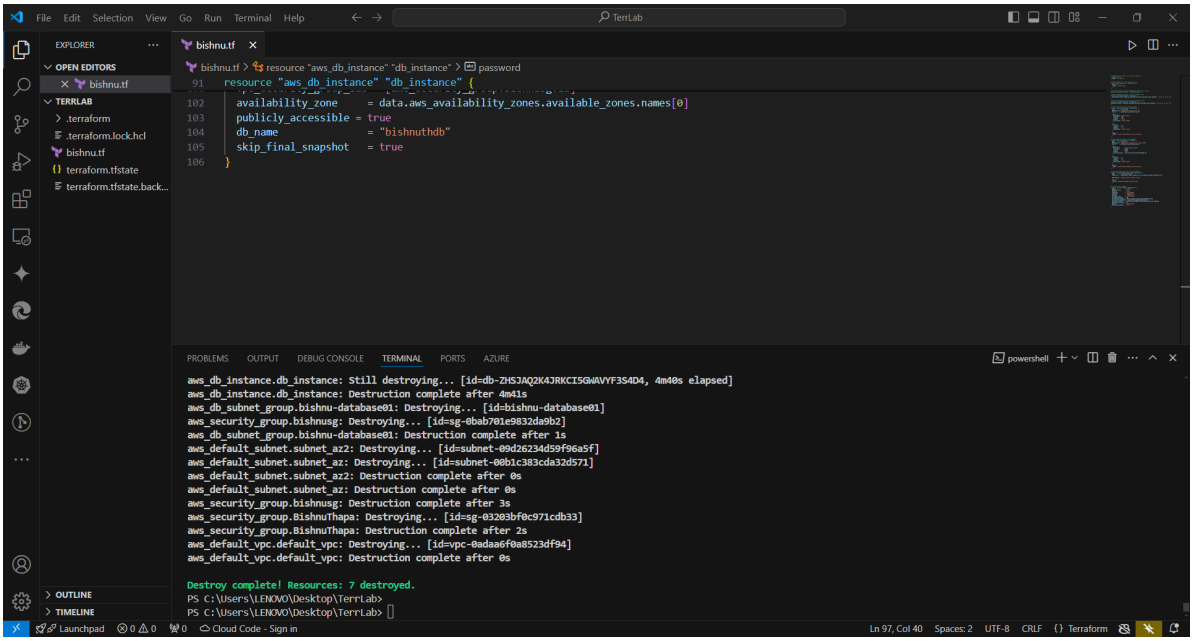


It shows destroying.

We can verify that our RDS is deleting as follows:

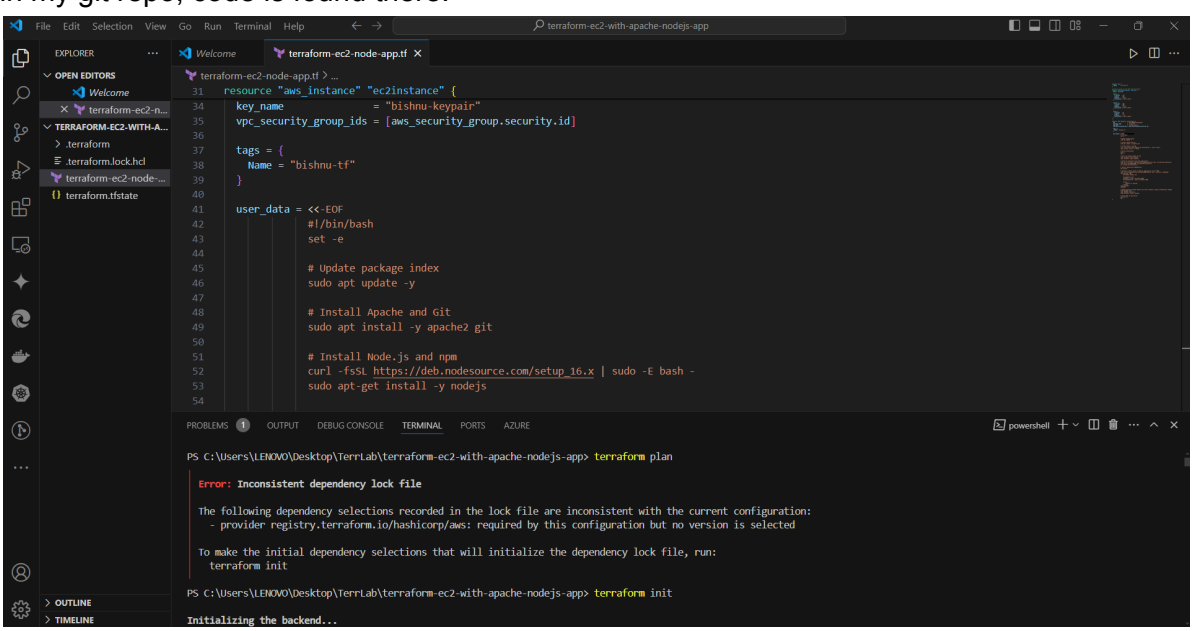


After finally successful destruction of resource, we get output look likes:

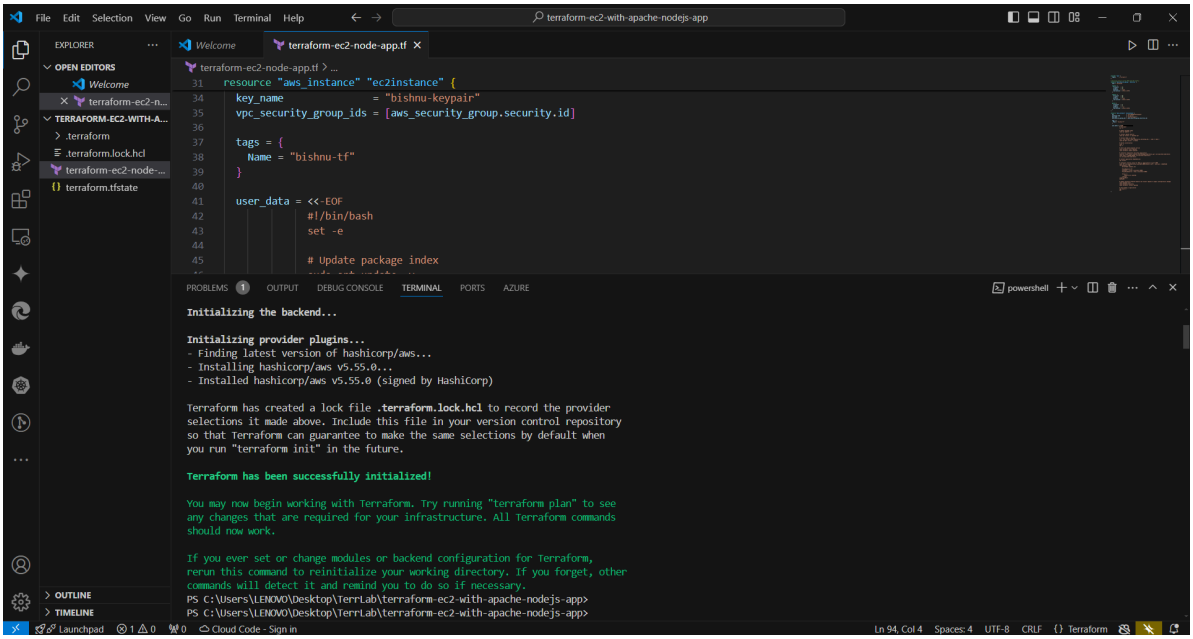


EC2 with apache2 webserver and nodejs sample app using terraform:

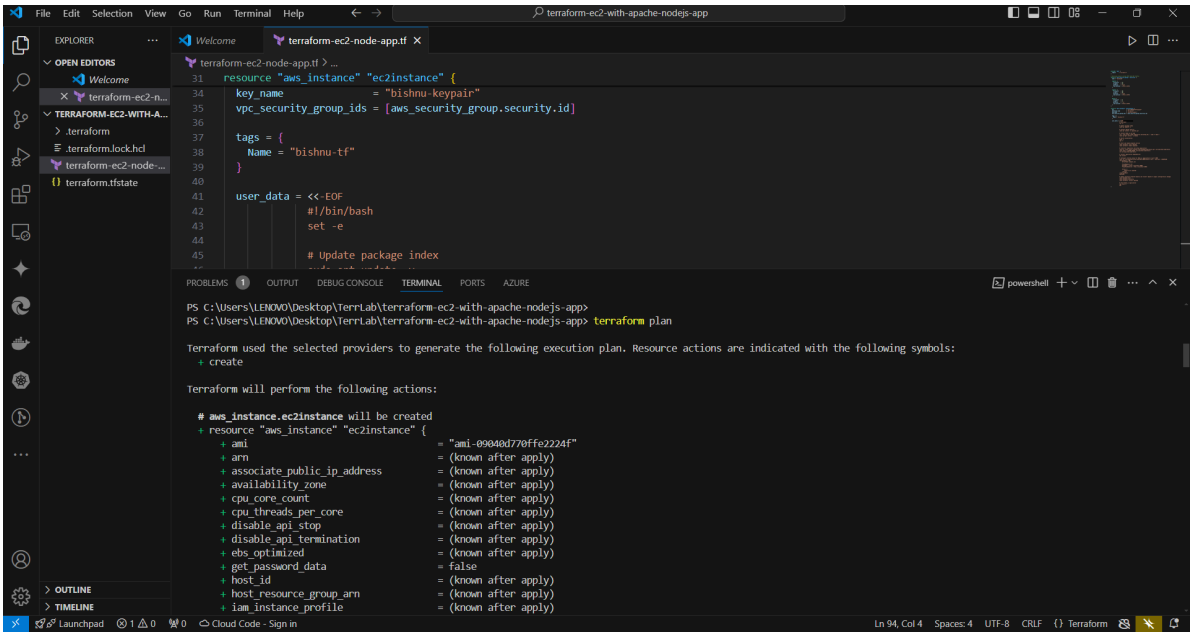
So, RDS is working, let’s deploy our node app which resides in GitHub repo, provided code in my git repo, code is found there:



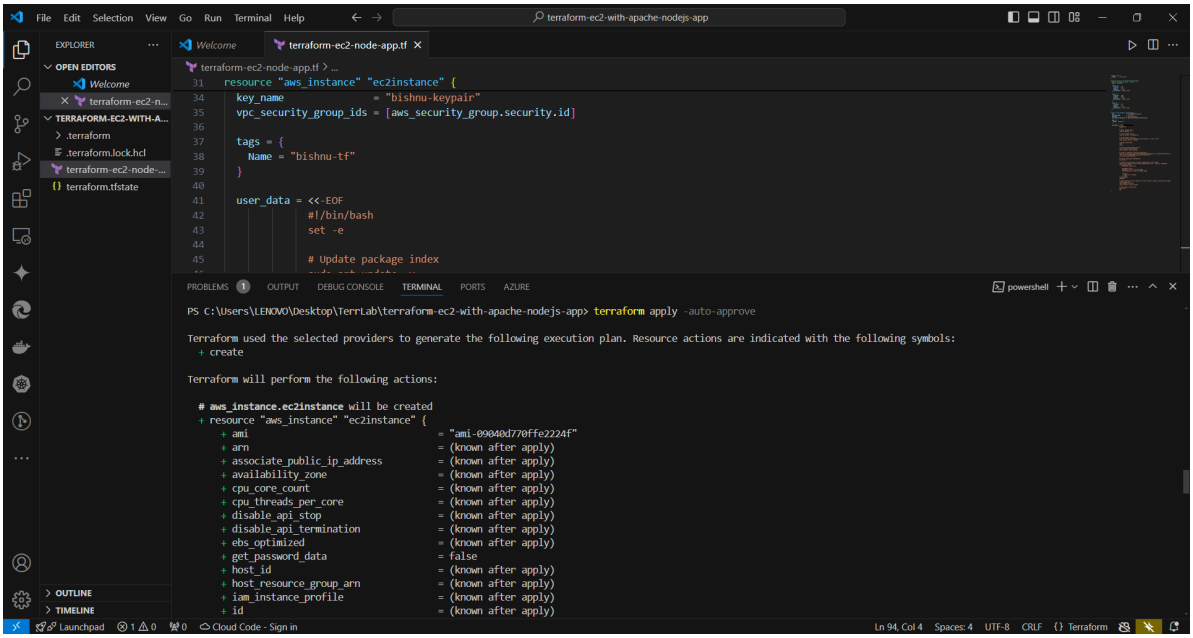
Let’s initialize out terraform in its folder.
Terraform initialized properly:



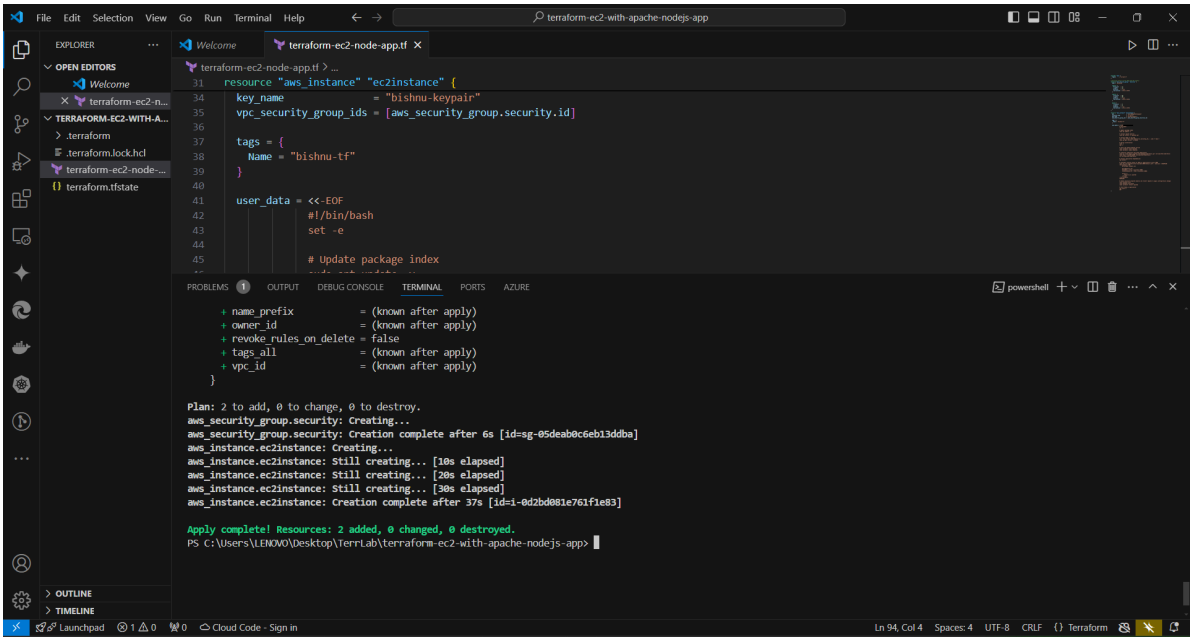
Run commands terraform plan, terraform validate and terraform apply simultaneously:



Commands are explained first step. After command executed, our infra will be provisioned.

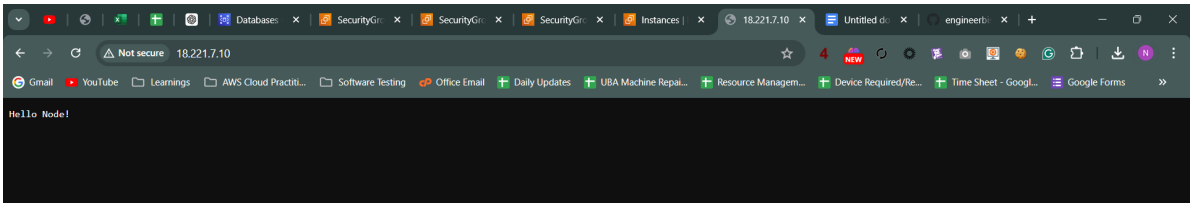


Here, resource is provisioning with green + sign there.

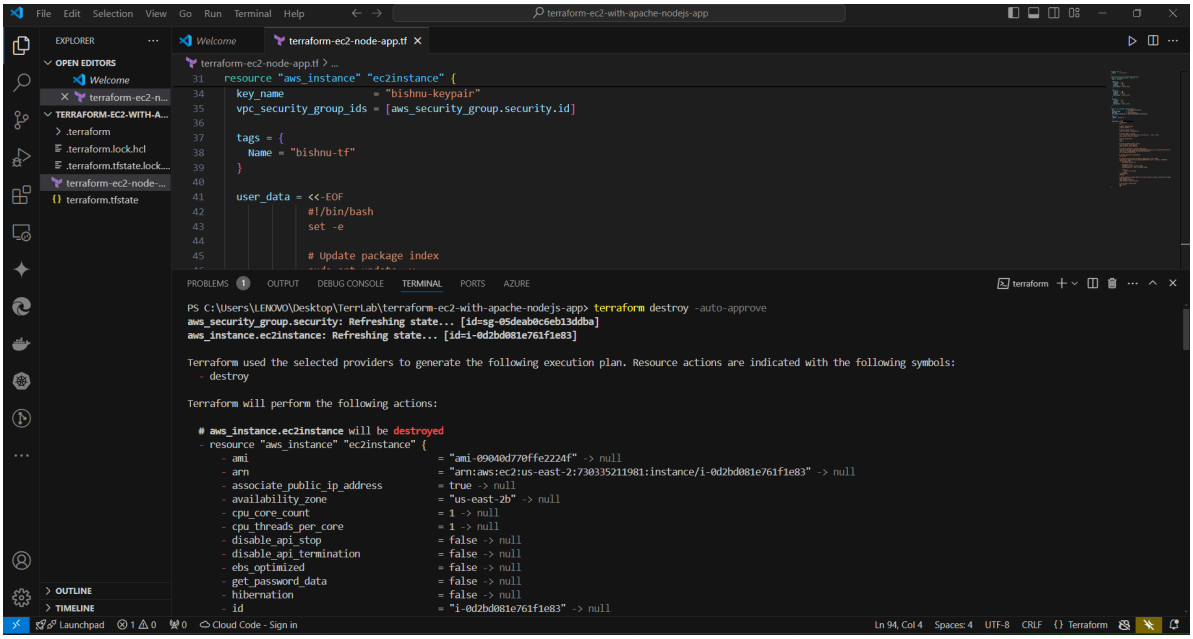


Finally resource created, ec2 has been running.

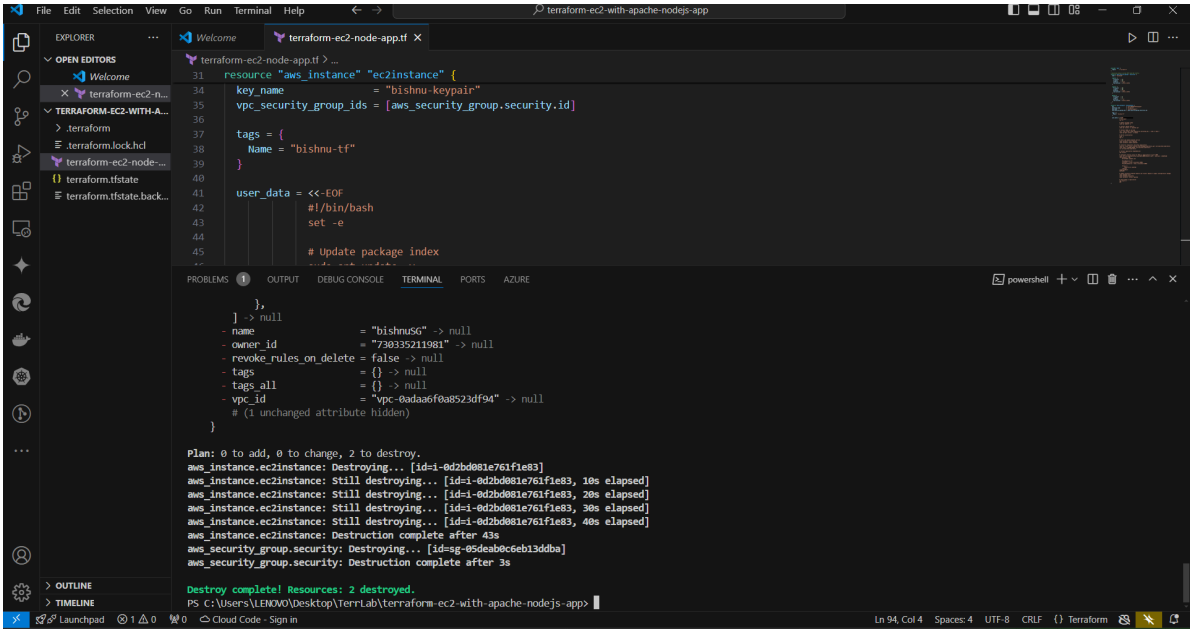
After sometimes of ec2 warmup, we can browse our ec2's public ip and there our node app is running and interface shows:



Destroying our EC2:



Now successfully destroyed:



Now, we have destroyed our infrastructure.

Some commands for working with github repos

Step-by-Step Instructions:

Cloning the Repository:
Navigate to the directory where you want to clone the repository (like my-terraform). Then, run the following command to clone the repository from GitHub:

```
git clone https://github.com/engineerbishnu/my-terraform.git
cd my-terraform
```

This command will create a new directory named my-terraform in your current directory (if it doesn't exist already) and clone the contents of the GitHub repository into it.

Create a New Branch:
Once inside the my-terraform directory, you can create a new branch named 2024-06-21/terraformlab1 and switch to it using the following command:

```
git checkout -b 2024-06-21/terraformlab1
```

Add and Commit Your Files:
Now, add your local files to the repository. Make sure you are in the my-terraform directory and replace <file> with the actual file or folder names you want to add or alternatively, to add all files and folders, use:

```
cd my-terraform  # Ensure you are in the repository directory
git add <file1> <file2> ...
git add .
```

Commit the changes with a meaningful commit message:

```
git commit -m "Initial commit for terraformlab1"
```

Push the Branch to Remote:
Finally, push the newly created branch 2024-06-21/terraformlab1 to the remote repository (origin), which is hosted on GitHub:

```
git push origin 2024-06-21/terraformlab1
```

Now verify on GitHub:
After pushing the branch, visit your GitHub repository (https://github.com/engineerbishnu/my-terraform) in browser.