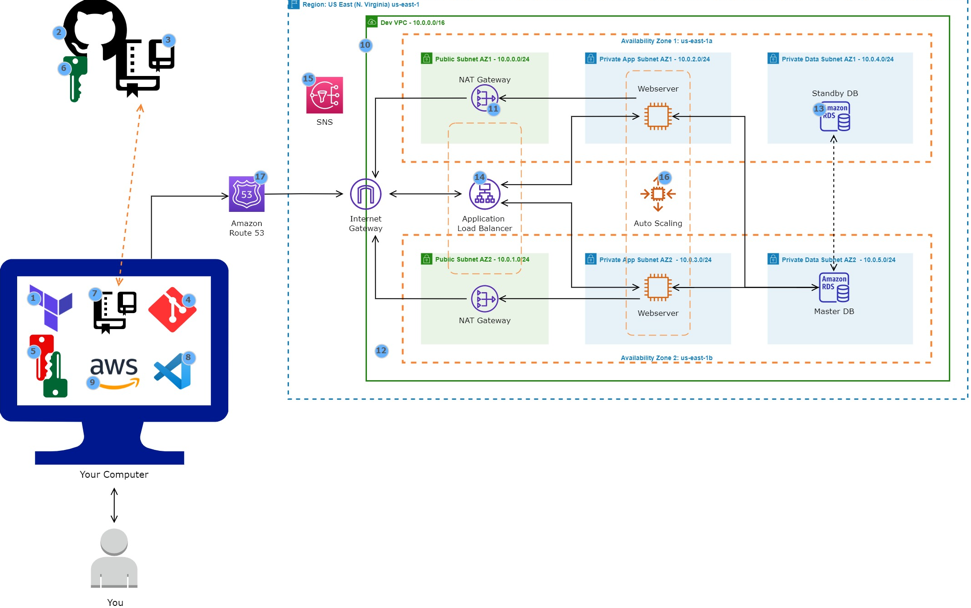
Host a Dynamic Ecommerce Website on AWS with Terraform



Resources:

1. Create Key Pairs on your computer
2. Add the Public SSH Key to GitHub to connect to GitHub without supplying your username and personal access token at each visit
3. Create VPC with public and private subnets in 2 availability zones
4. Create the Nat Gateways to allow the instances in the private App subnets and private Data subnets to access the internet
5. Create the Security Groups to open the required ports
6. Create an Application Load Balancer to distribute web traffic across an Auto Scaling Group of EC2 instances in multiple AZs
7. Create an SNS Topic to notifications
8. Create an autoscaling group to dynamically create your EC instances to make your website highly available
9. Create a record set in Route 53

DevOps Tools:

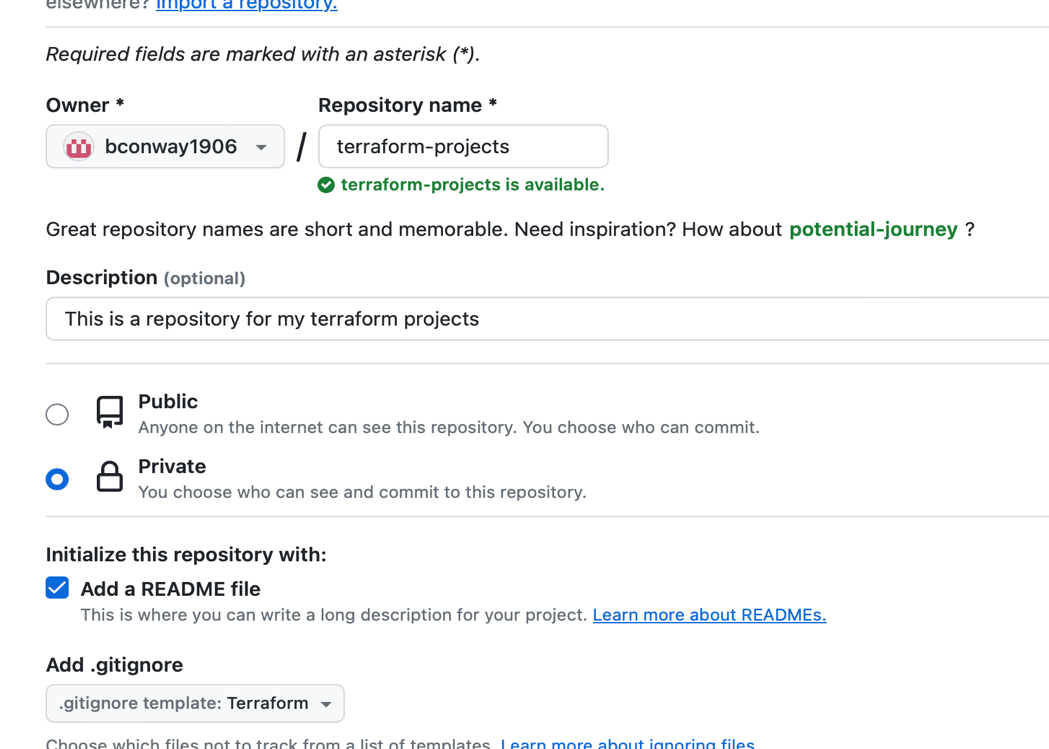
1. Install Terraform on your computer (infrastructure as code)
2. Sign up for a free GitHub account to work on your projects from anywhere
3. Create a GitHub Repository to store your codes
4. Install Git on your computer for source code management
5. Clone your GitHub repository to your computer to easily create and edit your terraform files
6. Install Visual Studio Code on your computer. VS Code is a text editor
7. Install AWS CLI on your computer to manage your AWS services
8. Create a MYSQL RDS database

Install Terraform on Mac

1. First install homebrew
2. Then on homebrew site, search for terraform then install

Create GitHub Repository

1. Name is terraform projects
2. Select .gitignore with terraform template

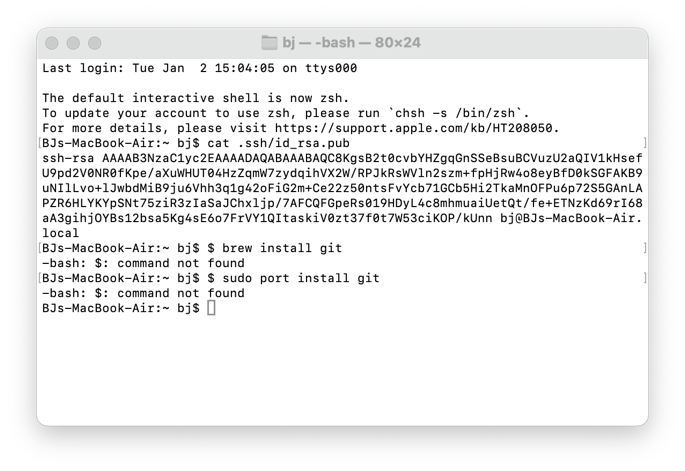


Create Keypair on Mac:

1. Open terminal and run the following command under your username:
   1. “Ssh-keygen -t rsa”
   2. Enter the file in which you wish to save the key
   3. Enter passphrase (not recommended to use a passphrase)
   4. Once the final message appears, run the following command to view the new private and public key: “ls -1 ~/.ssh/”

Add the public SSH Key to Github:

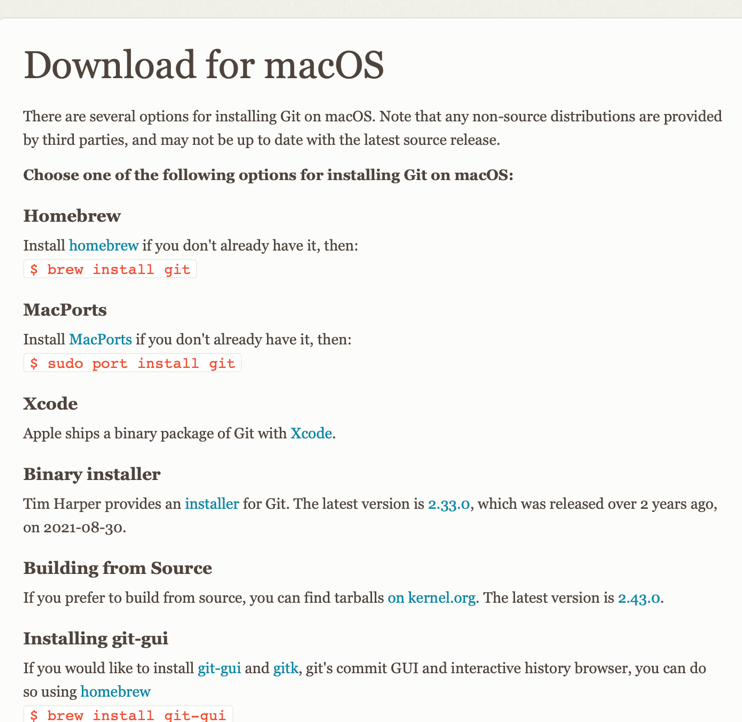
1. Run the following command in terminal: “cat .ssh/id\_rsa.pub”. This will reveal the public key that needs to be uploaded to Github.



1. Login in Github and select account settings. Select SSH and GPG Keys on the left side, then select “new SSH key.” Give it a name, then paste the public key.

To clone a Github repository, you must install git on the computer.

1. <https://git-scm.com/downloads>



1. Run following command to make sure git is installed: Git --version
2. Now configure username and email with the following command:
   1. git config --global user.name "Your Name"
   2. git config --global user.email "youremail"

Clone terraform-projects repository

1. Open terminal and change directory where the clone repository will be saved. I’m saving it to the desktop. Run command: cd desktop.
2. Run command: git clone [git@github.com:bconway1906/terraform-projects.git](mailto:git@github.com:bconway1906/terraform-projects.git)

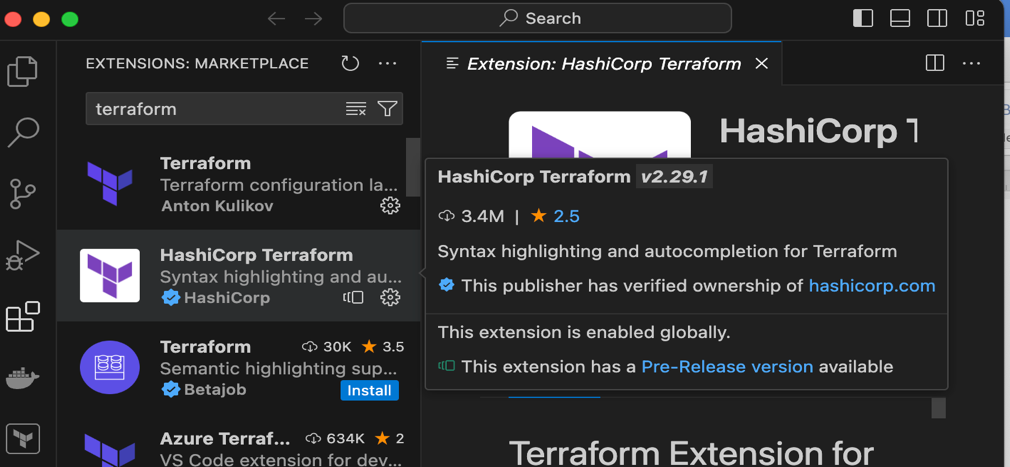
Push Changes from Local Repository to Remote Repository

1. Open terminal and make sure you are in the directory where the clone repository is
2. Run command: git add -A (to stage all changes)
3. Run command: git commit -m “first commit” (commits the changes)(m stands for message)
4. Run command: git push (this pushes all changes made on local computer to repository)

Install Visual Studio Code

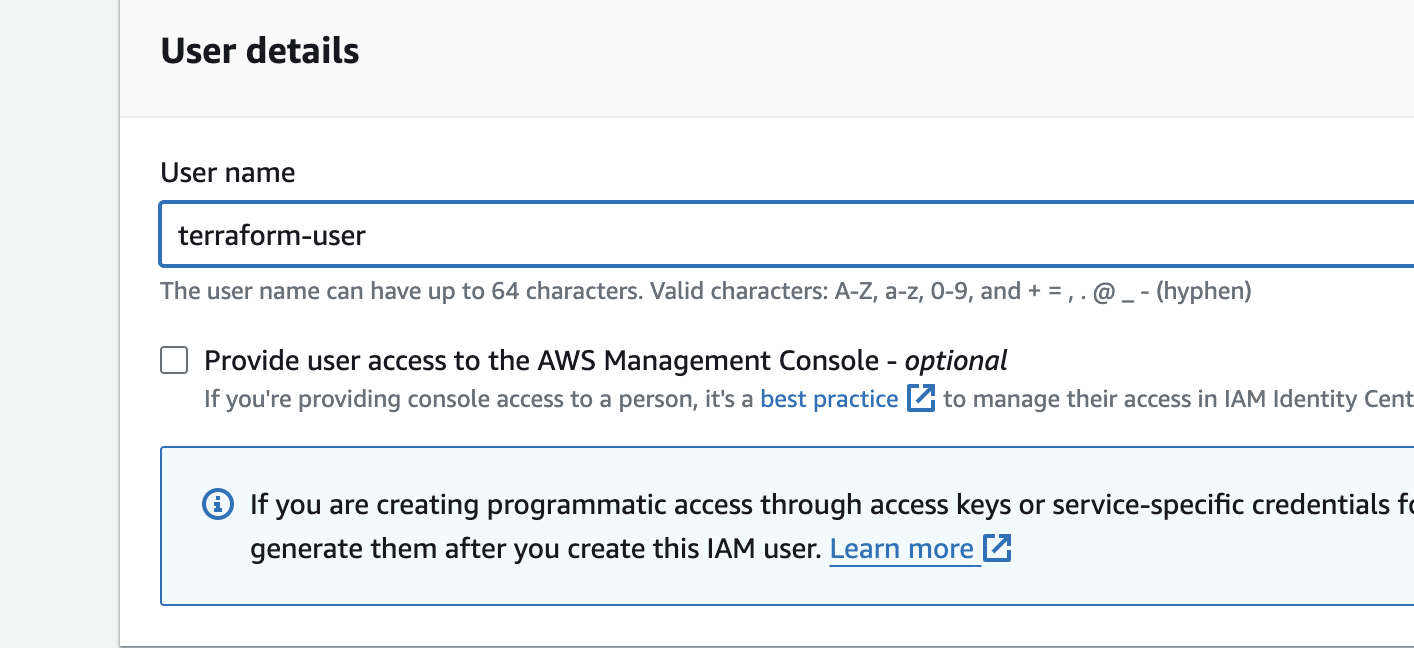
Install the extensions for Terraform

1. Open visual studio code
2. Select the extensions icon on the left
3. Download and install



Install the AWS Command Line (CLI) on a Windows Computer

Create IAM user AWS



1. Attach policy directly and give admin access

Generate access key for IAM user to have programmatic access

1. Use case: Command Line Interface
2. Copy access key and secret access key

Create a named profile for IAM User

1. This allows terraform to use the credentials to authenticate with AWS environment
2. In terminal, run command: aws configure --profile terraform-user (name if the IAM user)
3. Put in access key id
4. Secret key access
5. Then hit enter for none on default output

Create an S3 Bucket to Store the Terraform State File

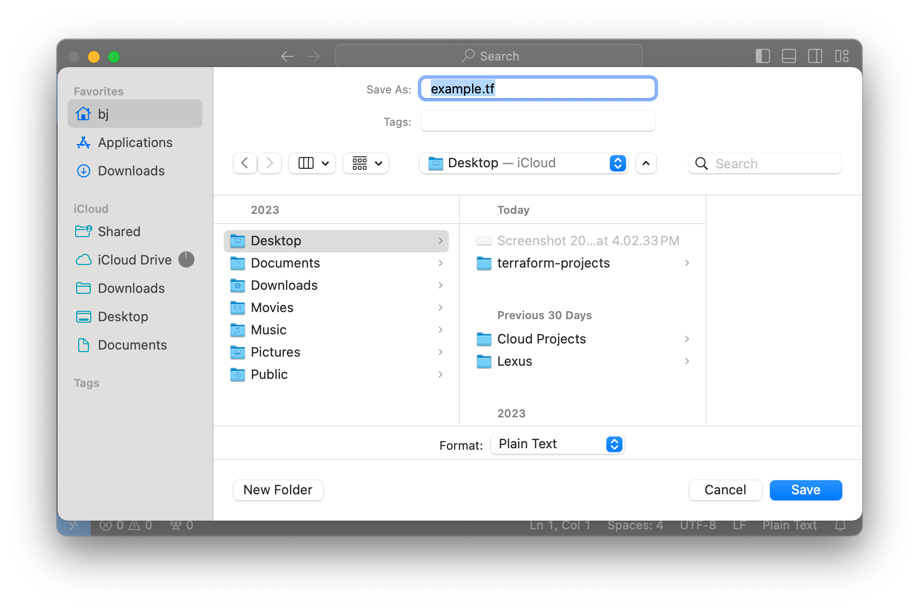
1. When you use terraform to create resources in AWS, it will record the information about the resources it created in a terraform state file. Next time you go to update those resources, terraform will use the state file to find those resources and update them accordingly.
2. Most companies save their terraform state file in an S3 bucket.
3. Enable bucket versioning

Locking Terraform State with Dynamic DB Table

1. This prevents multiple users from making changes to the state at the same time

How to Write a Terraform Syntax to Create any Resource in AWS

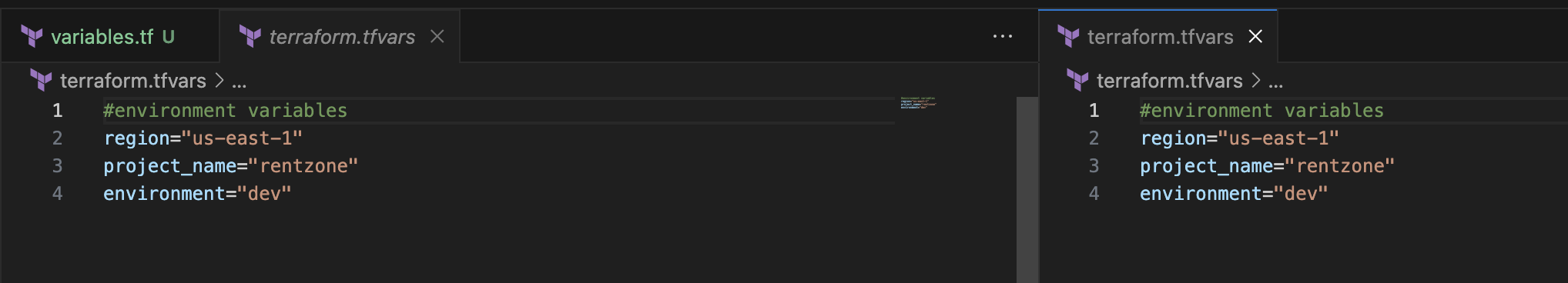
1. Open new file in visual studio code then save it with .tf extension to save in terraform format



1. When creating resources and terraform, you can google to find documentation.
2. When creating synax in terraform, it should always start with resource, then resource type(“aws\_vpc”). That’s how terraform knows what type of resource you are trying to create. Then local resource name (“vpc”) recommends use name as description you are trying to create. For every bracket you open, you need a closed bracket. Between brackets are arguments (the configuration of resources). If you are setting up resources in terraform and don’t know which arguments to use, go into the AWS console and create that resource/setting. Documents you may find on google may only give a few arguments and not all
3. “string” is a word. When you use that as your type.

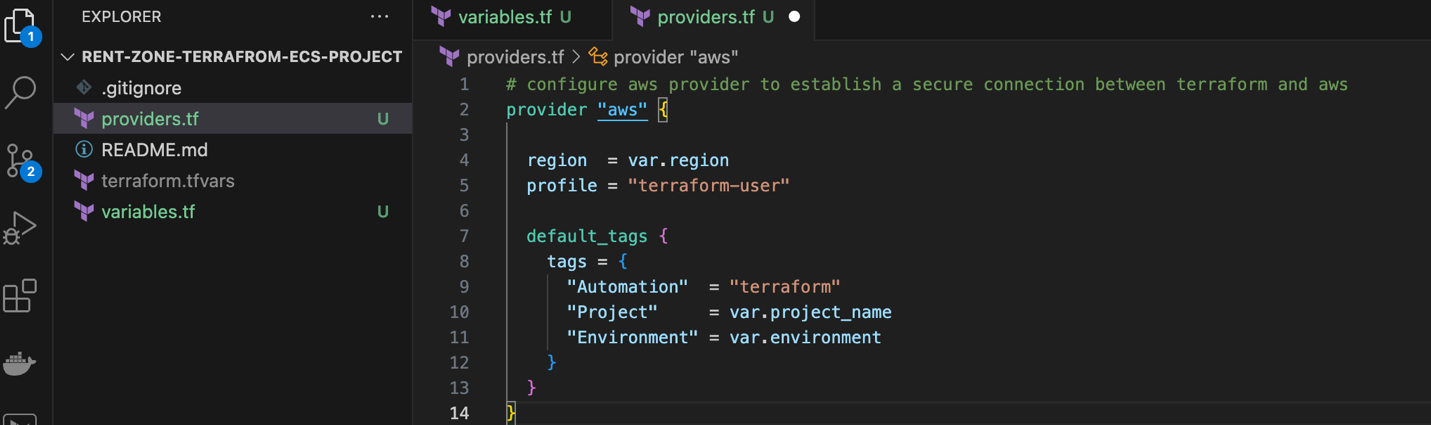
Assigning Values to Terraform Variables

1. Create a new file and name it “terraform.tfvars” this contains the values for the variables



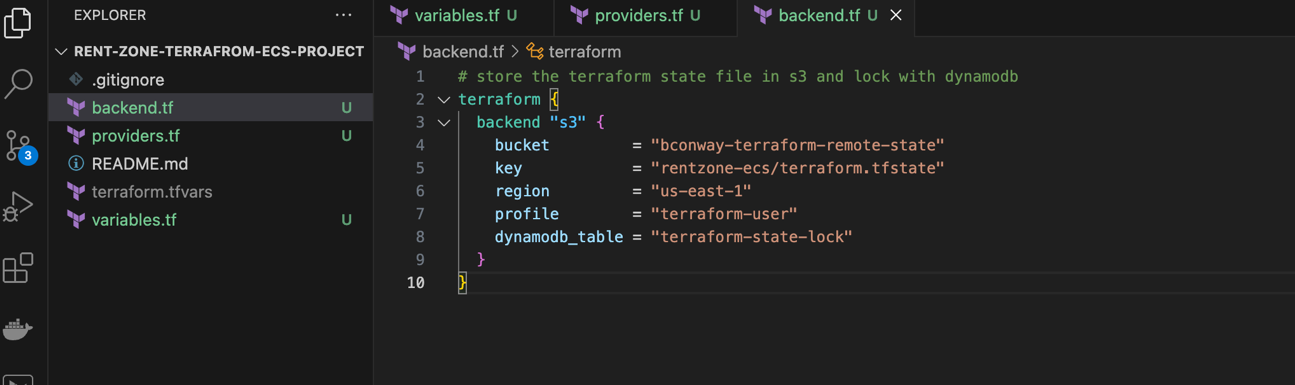
Establishing Secure Connection between Terraform and AWS

1. Open project folder in Visual Studio Code
2. Create a new file: providers.tf
3. Copy code from reference file file <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/providers-reference.tf>

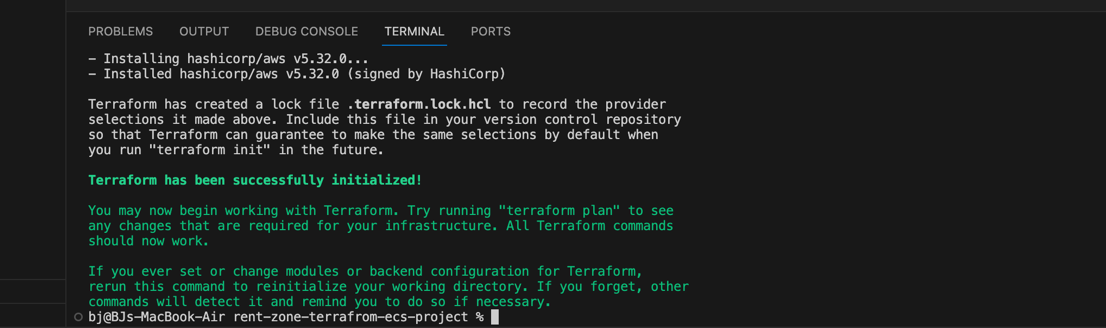


S3 Bucket and DynamoDB for Storing and Locking Terraform State

1. Copy reference file <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/backend-reference.tf>
2. Create new file “backend.tf”



1. Initialize with AWS environment
   1. Select open integrated terminal
   2. Check terminal shell vs code: select the drop-down arrow in the terminal panel and select the shell that recognizes the terraform command.
   3. Run command: “terraform init”

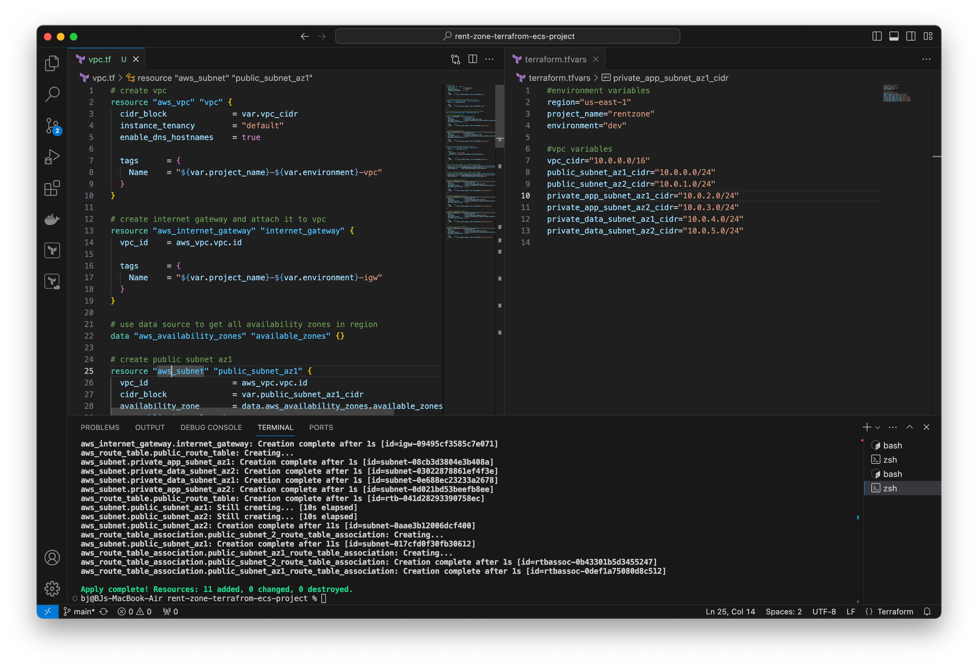


1. Push changes to the GitHub repository

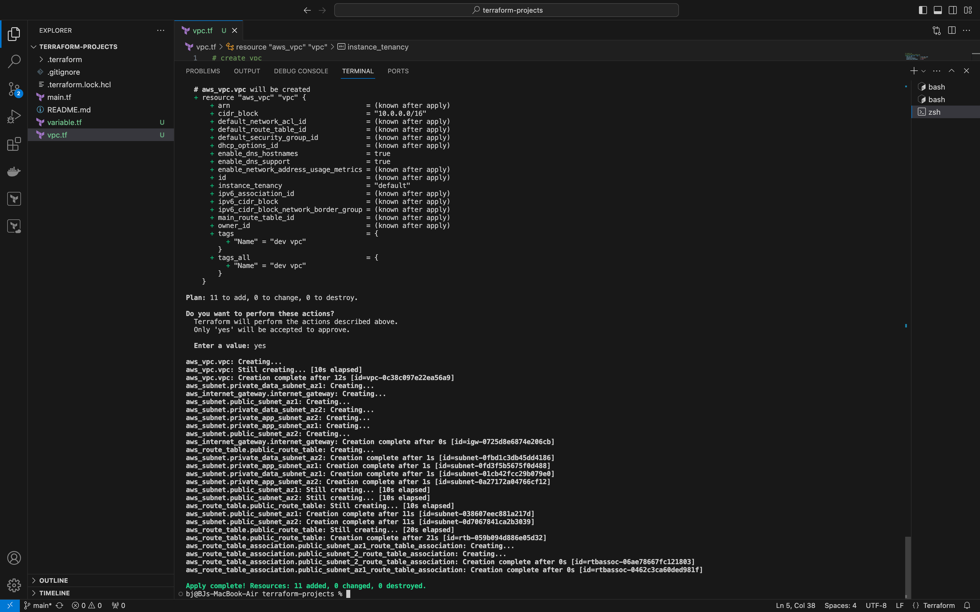
Create VPC with public and private subnets in 2 availability zones using terraform



1. Use reference file <https://github.com/azeezsalu/terraform-project-reference/blob/8b467f78170e23bcc63a9d5cc9f832c4fdb9fe7a/vpc%20reference.tf>
2. Fill in needed information in visual studio
3. Create new file vpc.tf, then paste sytax from reference file
4. Create a new file called variable.tf
5. When creating terraform syntax, the resource id is the resource type and local name you gave it and the attribute being referenced.
   1. Example: resource: “aws\_vpc””vpc”
   2. Example: vpc\_id: aws\_vpc.vpc.id
   3. When creating subnets. Use syntax “data.aws\_availability\_zones.available\_zones.names[0]” the zero is the first availability zone. Then change the number as you go down the list of what is available (ie [1])

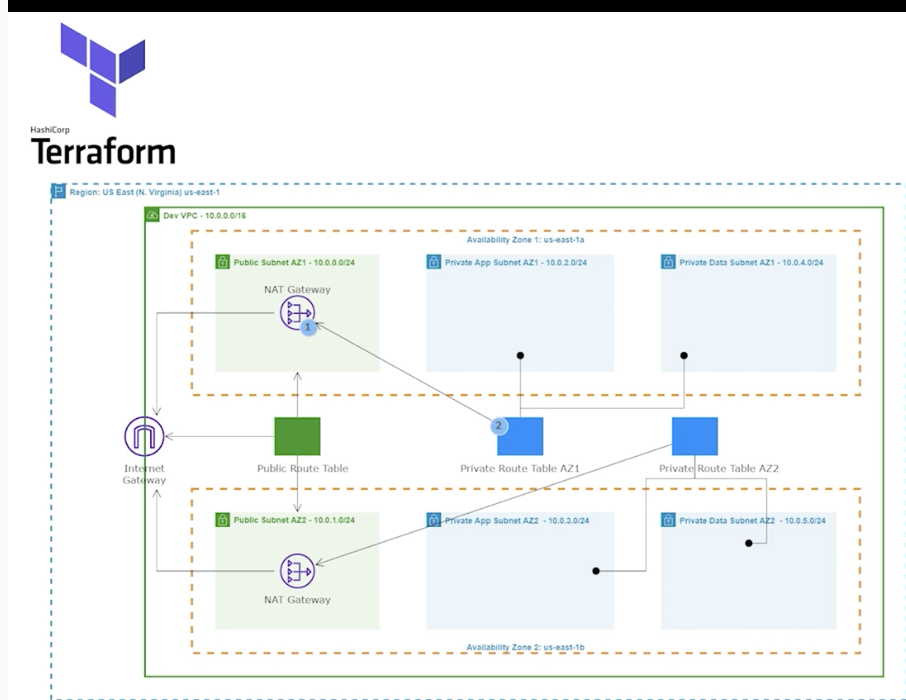


1. Close variable file
2. Use terraform to create VPC in AWS account
   1. Typing command “terraform plan” shows you what terrafrom will create in the AWS Account
   2. Once the plan is reviewed and is correct, run command: “terraform apply”



* 1. Push changes to repository
  2. Use command: “terraform destroy” to delete resources when finished.

Create Nat Gateway in public subnets AZ1 and AZ2. Nat Gateway allows the instances in the private app subnets and private data subnets to access the internet. Private route table is associated with the private subnets and routes traffic to the internet through the Nat gateway.



1. Open terraform-projects folder.
2. Create a new file “nat-gateway.tf”
3. Download reference file https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/nat-gateway-reference.tf
4. Use terraform to create NAT Gateway in AWS account
   1. Always begin by typing command “terraform plan” to show you what terrafrom will create in the AWS Account
   2. Once the plan is reviewed and is correct, run command: “terraform apply”

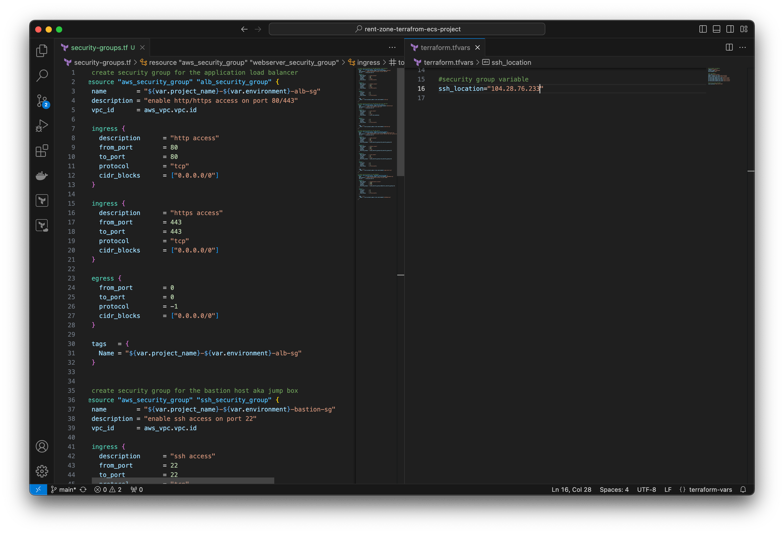


1. Push changes to repository

Create Security Groups using terraform

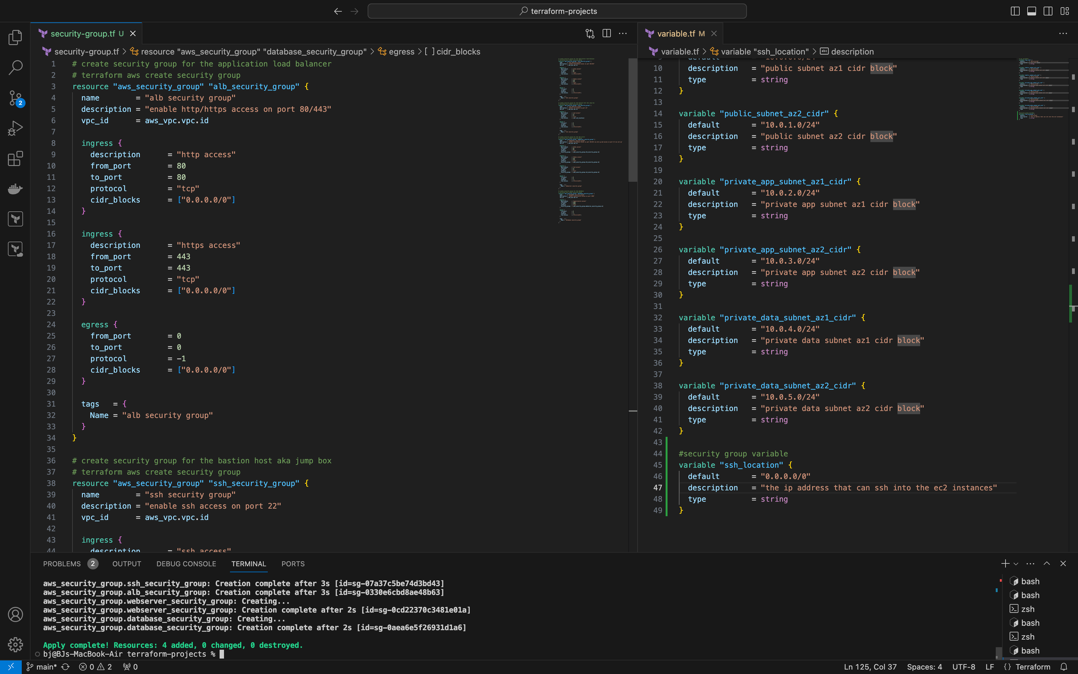


1. Application Load Balancer: Port = 80 and 443, Source = 0.0.0.0/0
2. SSH Security Group: Port 22, Source = Your IP Address
3. Webserver Security Group: Port = 80 and 443, Source = Application Security Group. Port = 22, Source = SSH Security Group
4. Database Security Group Port = 3306, Source = Webserver Security Group
5. Use reference <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/security-group-reference.tf>

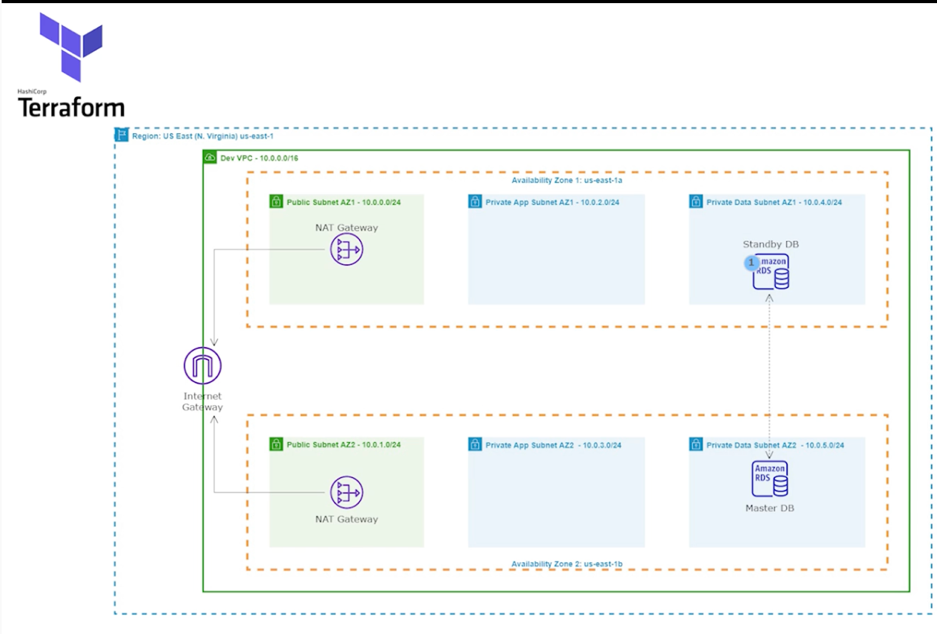


Use terraform to create security group in AWS account

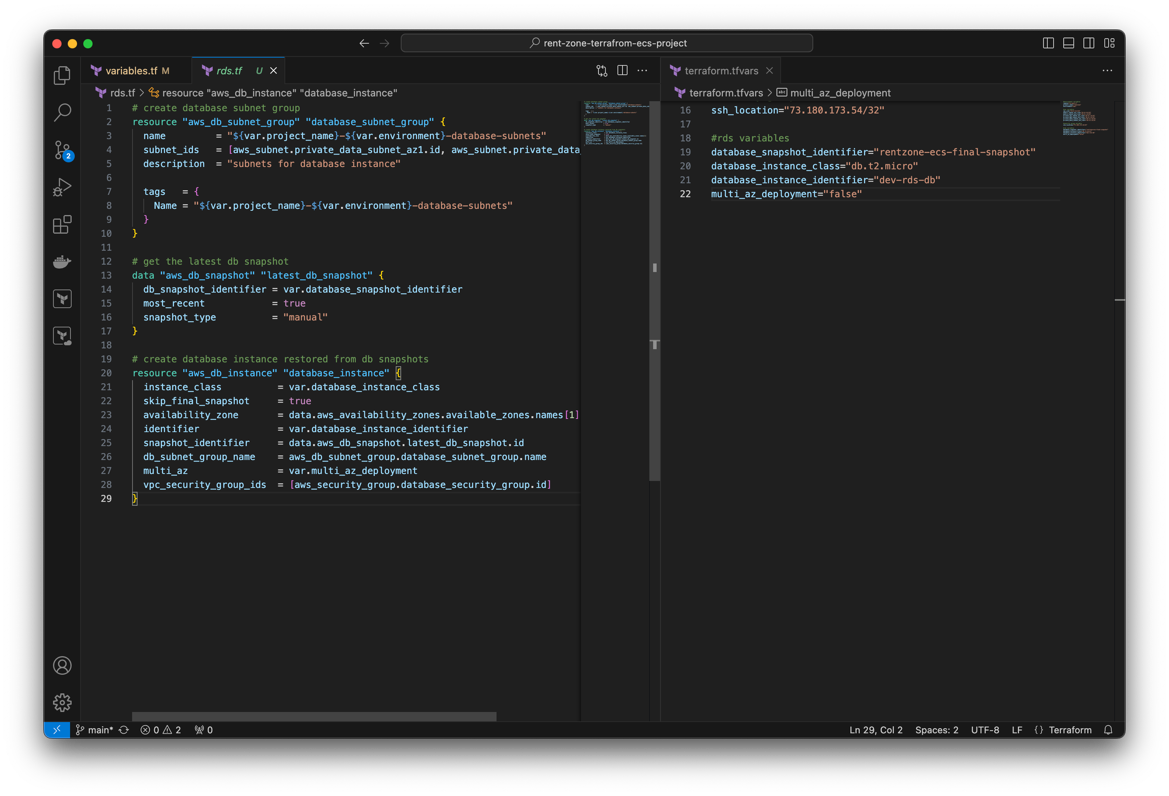
* 1. Always begin by typing command “terraform plan” to show you what terraform will create in the AWS Account
  2. Once the plan is reviewed and is correct, run command: “terraform apply”



Create RDS database by restoring it from a snapshot



1. Create new rds.tf file in Visual Studio Code
2. Download reference file <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/rds-reference.tf>
3. Use terraform to rds in AWS account
   1. Always begin by typing command “terraform plan” to show you what terraform will create in the AWS Account
   2. Once the plan is reviewed and is correct, run command: “terraform apply”



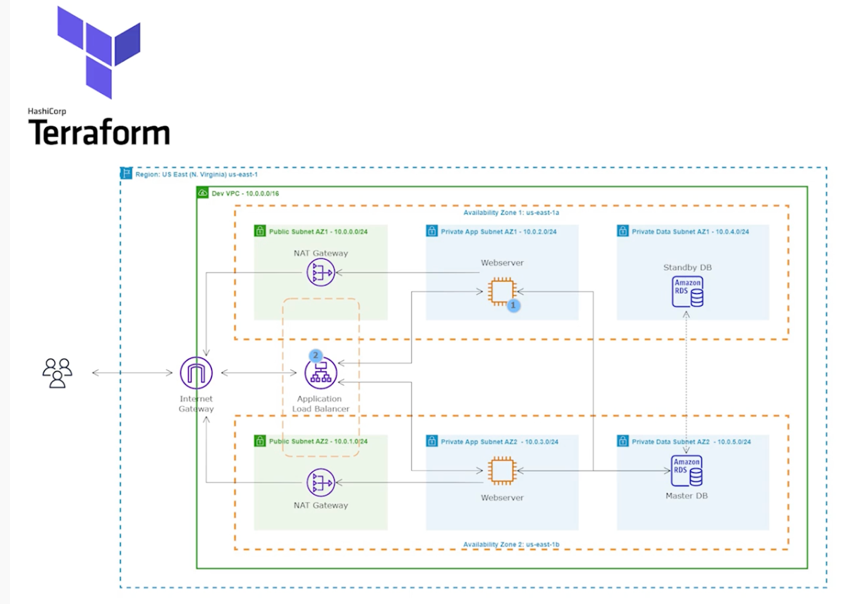
1. Push changes to repository

Create SSL certificate using Terraform

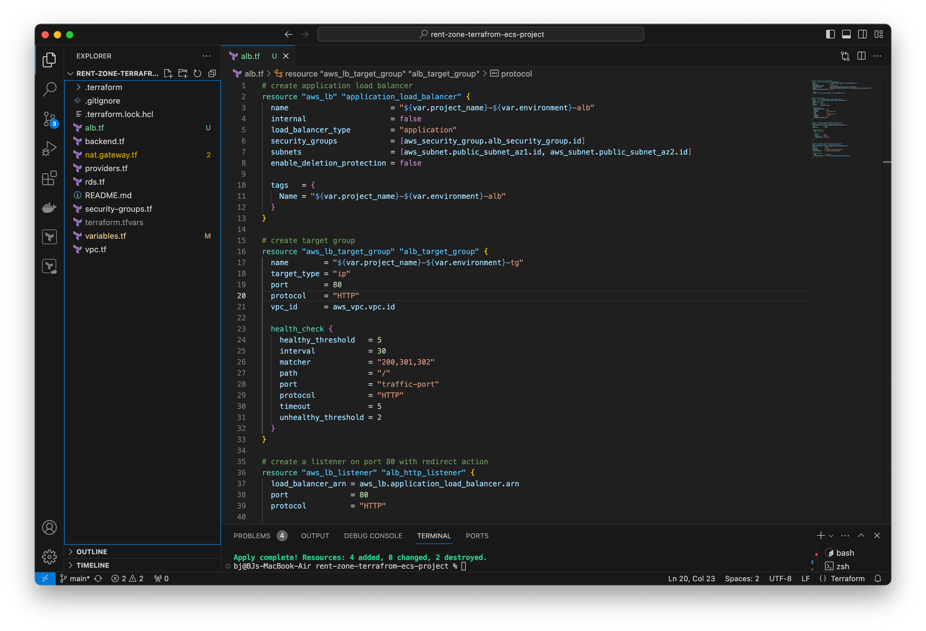
1. Open reference file <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/acm-reference.tf>



Create Application Load Balancer using Terraform to route traffic to the webservers in the private subnet



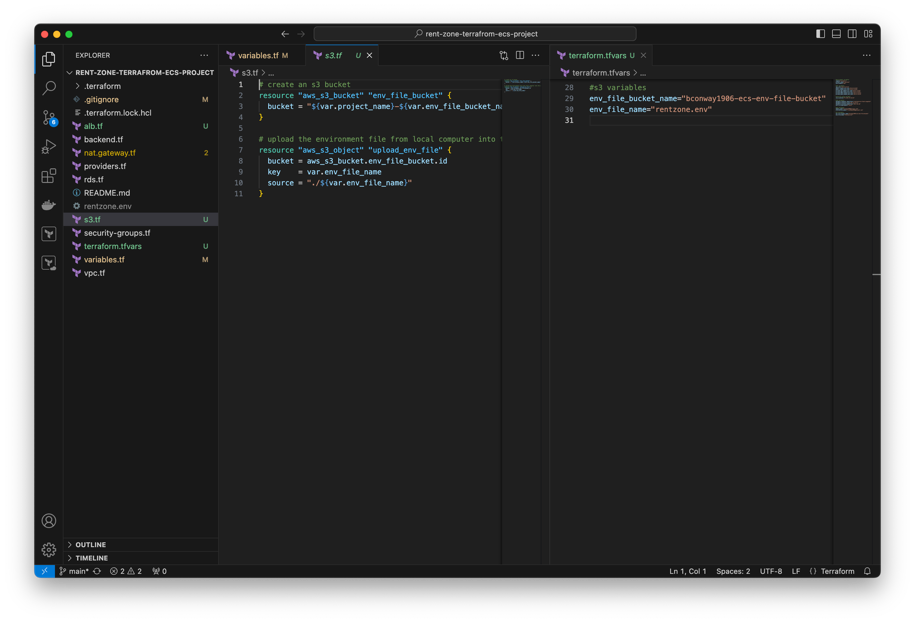
1. Open new file under terraform project “alb.tf”
2. Open reference file https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/alb-reference.tf
3. ALB will be internet facing so for “Internal” in the terraform syntax, type false
4. Target type will be “ip” which is different from the previous project
5. Use terraform to add ALB in AWS account
   1. Always begin by typing command “terraform plan” to show you what terraform will create in the AWS Account
   2. Once the plan is reviewed and is correct, run command: “terraform apply”



1. Push changes to repository

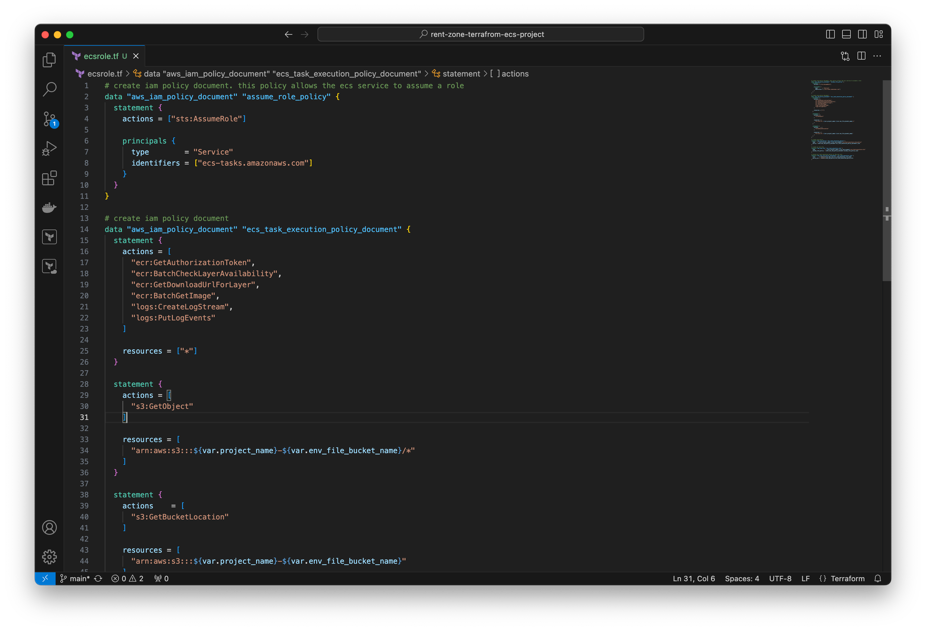
Create S3 Bucket with Terraform

1. Open reference file <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/s3-reference.tf>
2. Copy the rentzone.env file from the previous project and paste it into visual studio
3. Add the env file to gitignore
4. Create a new file “s3.tf”
5. Add variables definitions



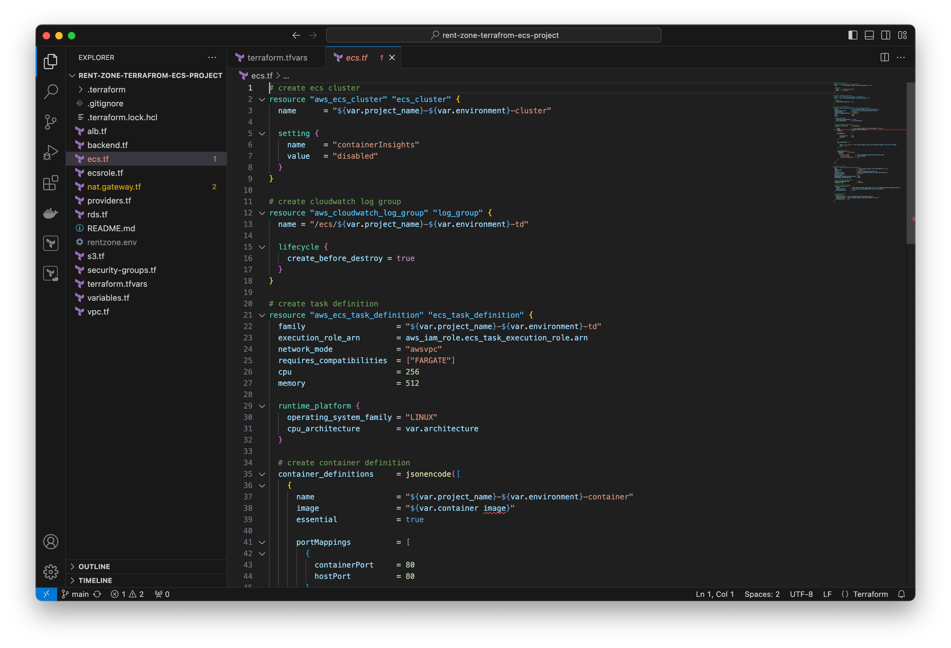
ECS Task Execution Role Creation with Terraform

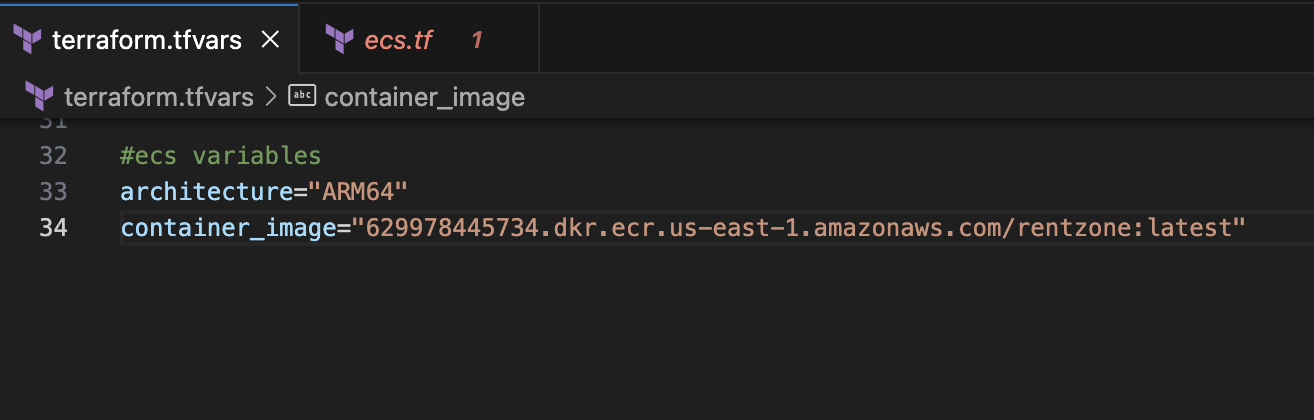
1. Create new file ecsrole.tf
2. Open reference file <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/ecs-role-reference.tf>



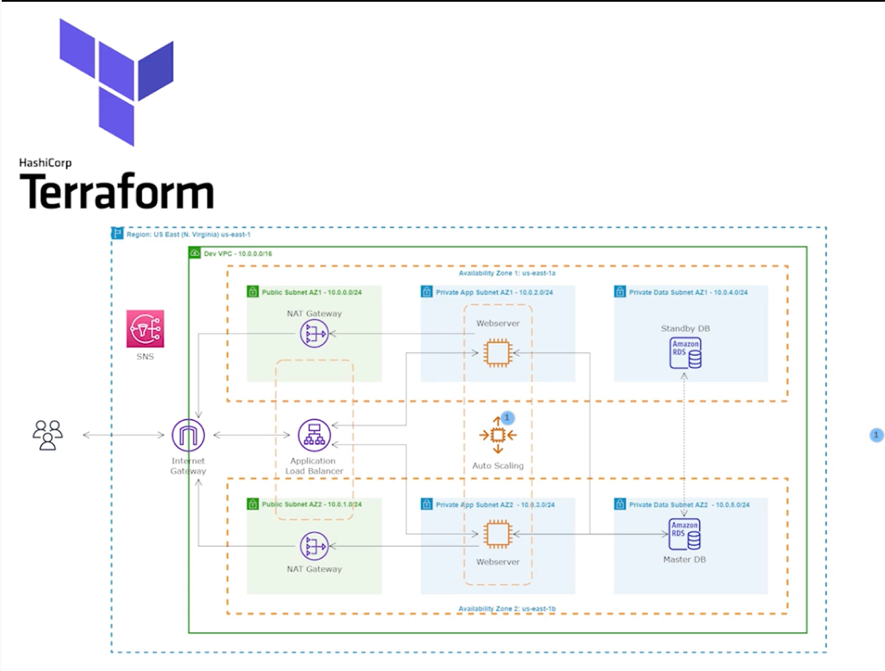
ECS Service Creation with Terraform

1. Open reference file <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/ecs-reference.tf>
2. Create new file ecs.tf
3. Note: if docker image was built on windows cpu\_architecture must be X86\_64, Mac must be ARM64

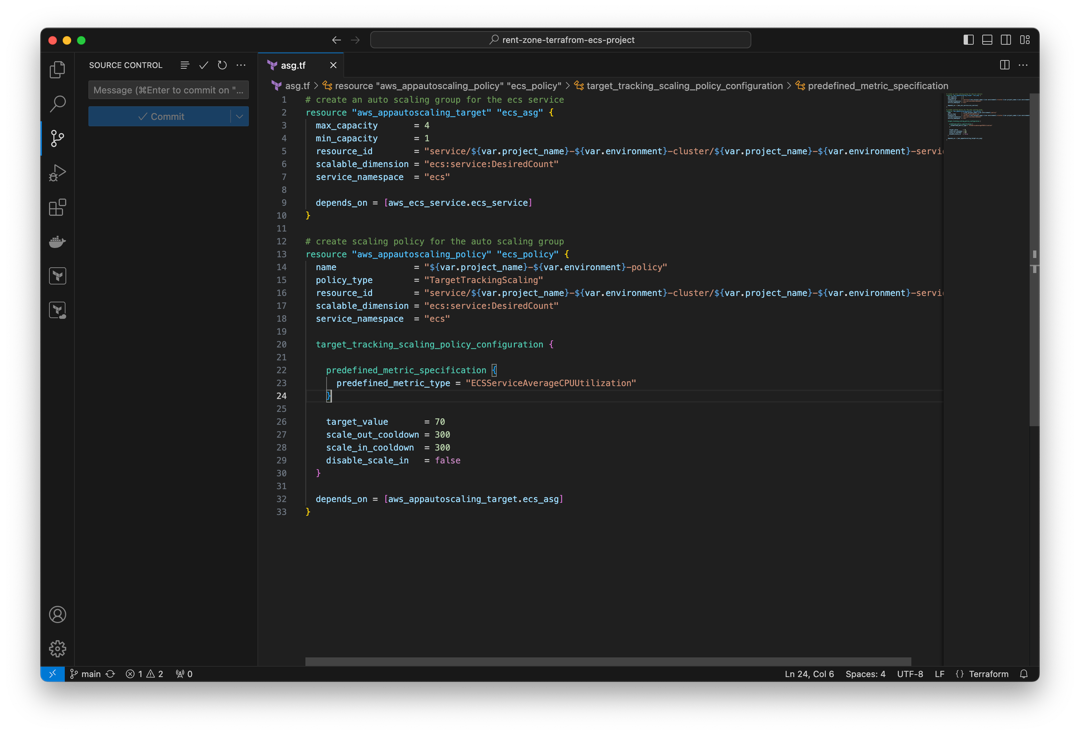




Create Auto Scaling Group with terraform. This will dynamically create our EC2 instances to make our website highly available, scalable, fault-tolerant, and elastic



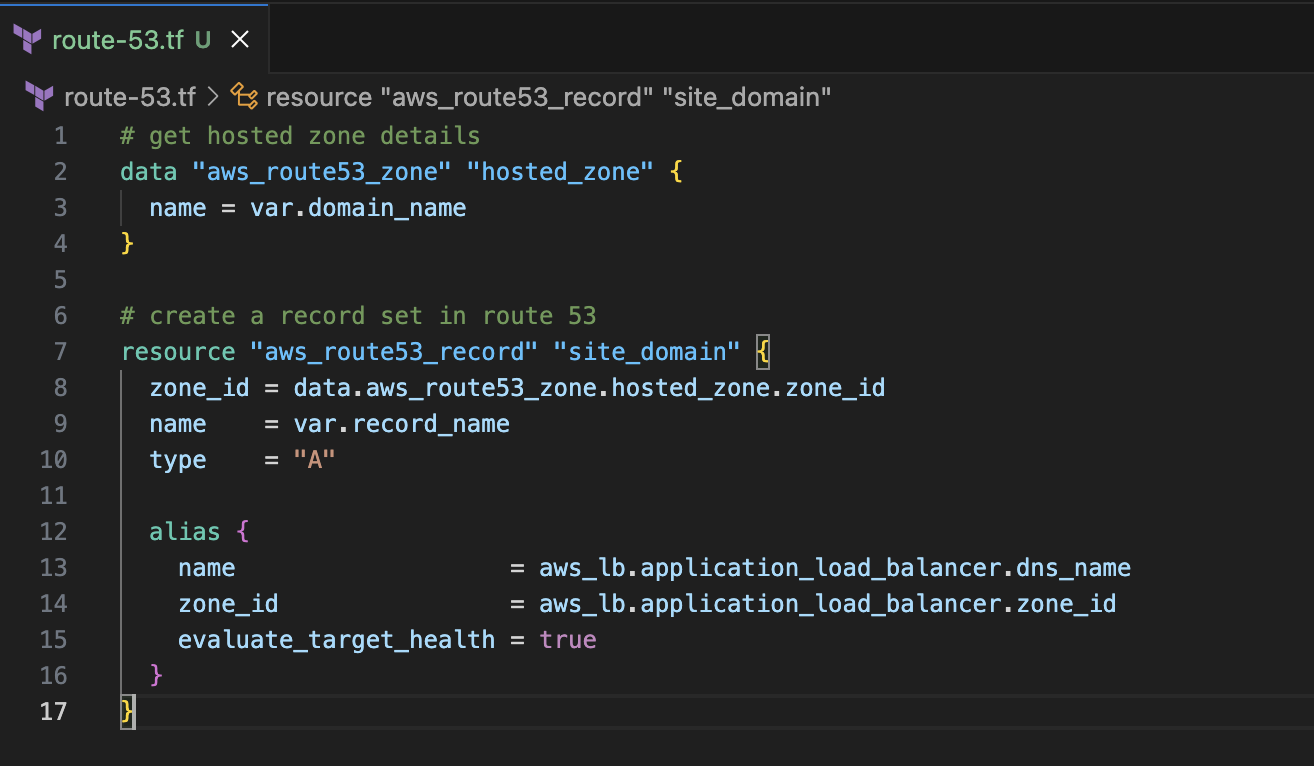
1. Open projects folder and create new file
2. Open reference file <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/asg-reference.tf>

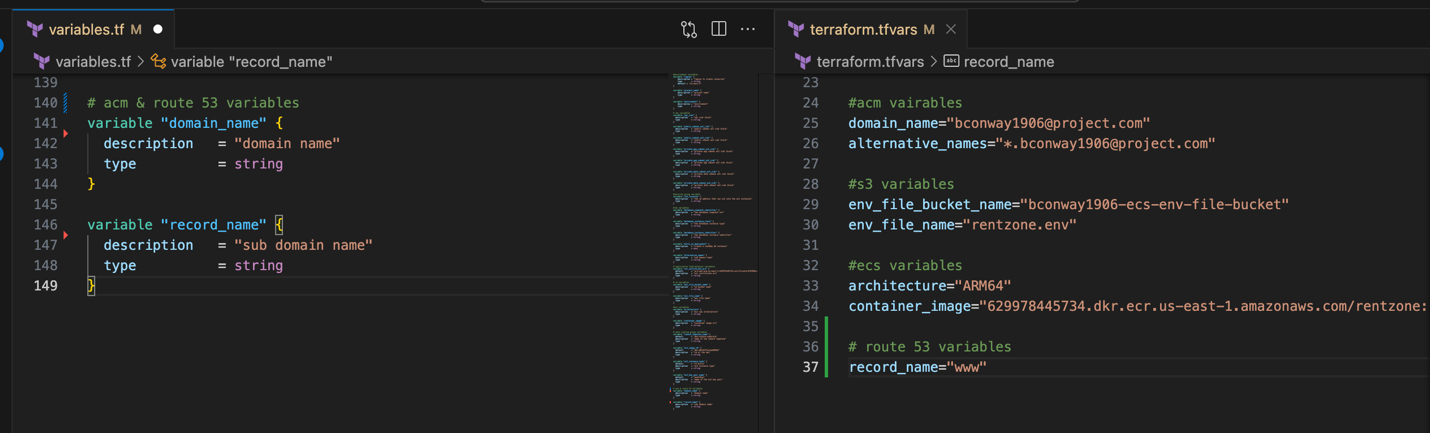


1. Push changes to repository

Create Record Set in Route-53 and Terraform Output

1. Open projects folder and create new file
2. Open reference file <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/route-53-reference.tf>

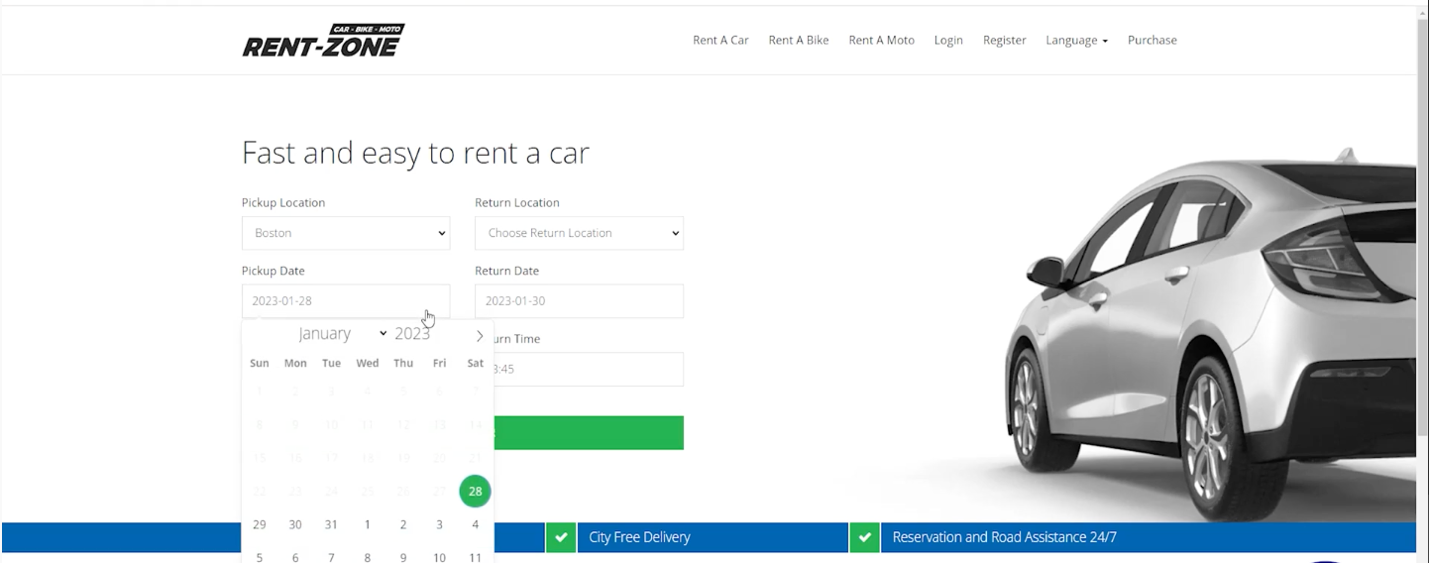




1. Open reference file <https://github.com/azeezsalu/aosnote-projects/blob/main/terraform-projects/rentzone-terraform-ecs-project/outputs-reference.tf>
2. Create new file outputs.tf



Completed website:



Clean Up

1. In terminal, type “terraform destroy”