**rentzone-infrastructure-ecs repository:** https://github.com/Silas-cloudspace/rentzone-infrastructure-ecs

**terraform-modules repository:** https://github.com/Silas-cloudspace/terraform-modules

**Create and Clone a Repository for storing Terraform Modules**

Create a new repository in GitHub and call it “terraform-modules”

Make it private

Clone it to your desired location.

**Create Terraform Module for VPC**

Open VS Code and navigate to the “terraform-modules” directory

Create a new folder: mkdir vpc

Create 3 new files in the “vpc” folder: touch main.tf variables.tf outputs.tf

Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created vpc module”
* git push

**Create and clone a Terraform Infrastructure Code Repository**

Create and clone to your computer a new repository for the project that we will use to deploy our application.

I’ve named mine “rentzone-infrastructure-ecs”

**Create a S3 bucket and a Dynamodb table**

For storing the Terraform state in the S3 and for locking Terraform state in the Dynamodb

Go to VS Code and create a new folder and name it “remote\_backend” in the “rentzone-infrastructure-ecs” directory

* mkdir remote\_backend

Navigate to this new folder and create a new file: “backend”

* touch backend

Copy the terraform file from the shared repository into it.

Run: terraform init and terraform apply

**Create Terraform Provider**

Navigate to the “rentzone-infrastructure-ecs” directory

touch providers.tf variables.tf

Copy the terraform files from the shared repository into them.

**Create Terraform backend**

touch backend.tf

Copy the terraform file from the shared repository into it.

Run: terraform init

**Create a 3 tier VPC**

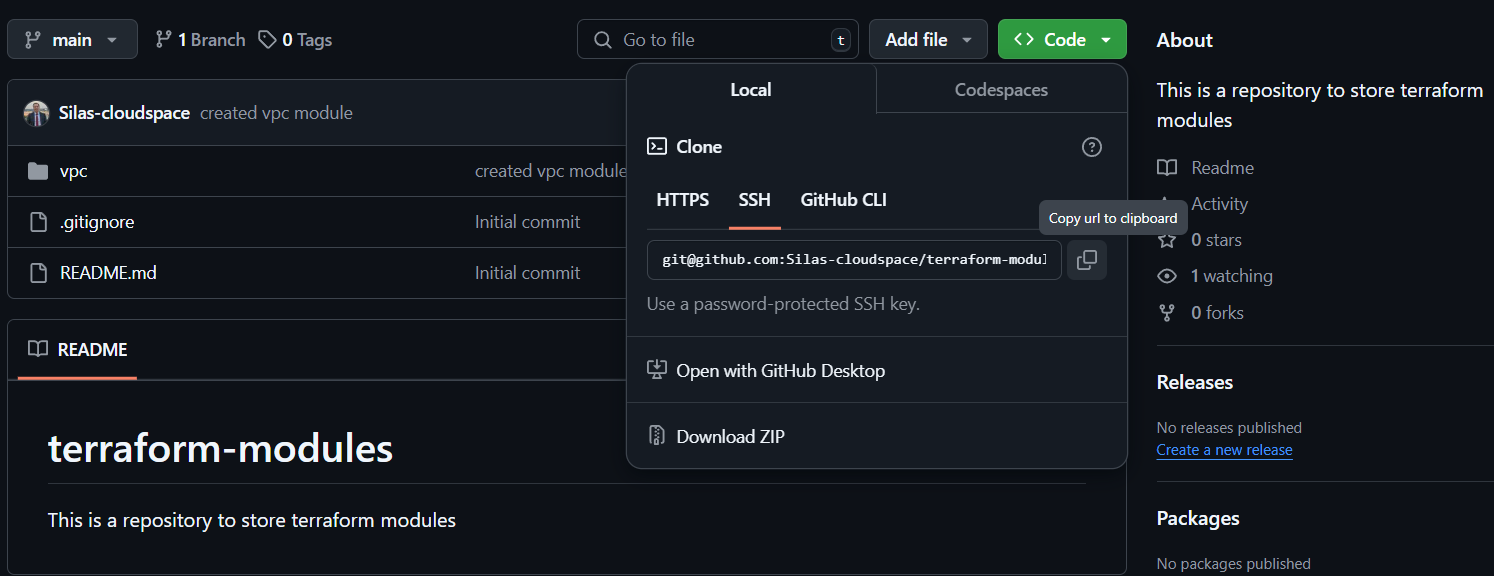
touch main.tf

Copy the terraform file from the shared repository into it.

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

We add //vpc to the end of the SSH URL because the module is located in the vpc folder

You need to do the same for the following modules that we create



touch terraform.tfvars

Copy the terraform file from the shared repository into it.

create a .gitignore.tf file and add the”terraform.tfvars” into it

**Important:** Your code will be bigger but run it only until the “module ‘vpc’”. Comment the rest of it by:

* **For Windows/Linux**: Press Shift + Alt + A
* **For macOS**: Press Shift + Option + A

Run:

* terraform fmt
* terraform init
* terraform apply

**Create Terraform Module for Nat Gateway**

Navigate to the “terraform-modules” directory

mkdir nat\_gateway

cd nat\_gateway

touch main.tf variables.tf

Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created nat gateway module”
* git push

**Create NAT Gateways for the VPC**

Navigate to the “rentzone-infrastructure-ecs” directory

Go to the main.tf file

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

Add //nat\_gateway to the end of the SSH URL because the module is located in the GitHub “nat\_gateway” folder we recently created

**Important:** Your code will be bigger but run it only until the “module ‘nat\_gateway’”. Comment the rest of it by:

* **For Windows/Linux**: Press Shift + Alt + A
* **For macOS**: Press Shift + Option + A

Run:

* terraform fmt
* terraform init
* terraform apply

**Create Terraform Module for the security groups**

Navigate to the “terraform-modules” directory

mkdir security\_groups

cd security\_groups

touch main.tf variables.tf outputs.tf

Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created security groups module”
* git push

**Create Security Groups for the VPC**

Navigate to the “rentzone-infrastructure-ecs” directory

Go to the main.tf file

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

Add //security\_groups to the end of the SSH URL because the module is located in the GitHub “security\_groups” folder we recently created

**Important:** Your code will be bigger but run it only until the “module ‘security\_group”. Comment the rest of it by:

* **For Windows/Linux**: Press Shift + Alt + A
* **For macOS**: Press Shift + Option + A

Run:

* terraform fmt
* terraform init
* terraform apply

**Create Terraform Module for the Database**

Navigate to the “terraform-modules” directory

mkdir rds

cd rds

touch main.tf variables.tf

Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created rds module”
* git push

**Create Database for the VPC**

Navigate to the “rentzone-infrastructure-ecs” directory

Go to the main.tf file

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

Add //rds to the end of the SSH URL because the module is located in the GitHub “rds” folder we recently created

**Important:** Your code will be bigger but run it only until the “module ‘rds”. Comment the rest of it by:

* **For Windows/Linux**: Press Shift + Alt + A
* **For macOS**: Press Shift + Option + A

Run:

* terraform fmt
* terraform init
* terraform apply

**Create Terraform Module for the ACM**

Navigate to the “terraform-modules” directory

mkdir acm

cd acm

touch main.tf variables.tf outputs.tf

Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created acm module”
* git push

**Request SSL certificate for your AWS Resources**

Navigate to the “rentzone-infrastructure-ecs” directory

Go to the main.tf file

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

Add //acm to the end of the SSH URL because the module is located in the GitHub “acm” folder we recently created

**Important:** Your code will be bigger but run it only until the “module ‘ssl\_certificate”. Comment the rest of it by:

* **For Windows/Linux**: Press Shift + Alt + A
* **For macOS**: Press Shift + Option + A

Run:

* terraform fmt
* terraform init
* terraform apply

**Create Terraform Module for the Application Load Balancer**

Navigate to the “terraform-modules” directory

mkdir alb

cd alb

touch main.tf variables.tf outputs.tf

Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created alb module”
* git push

**Create the Load Balancer**

Navigate to the “rentzone-infrastructure-ecs” directory

Go to the main.tf file

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

Add //alb to the end of the SSH URL because the module is located in the GitHub “alb” folder we recently created

**Important:** Your code will be bigger but run it only until the “module ‘application\_load\_balancer”. Comment the rest of it by:

* **For Windows/Linux**: Press Shift + Alt + A
* **For macOS**: Press Shift + Option + A

Run:

* terraform fmt
* terraform init
* terraform apply

**Create a private repository to store the application codes**

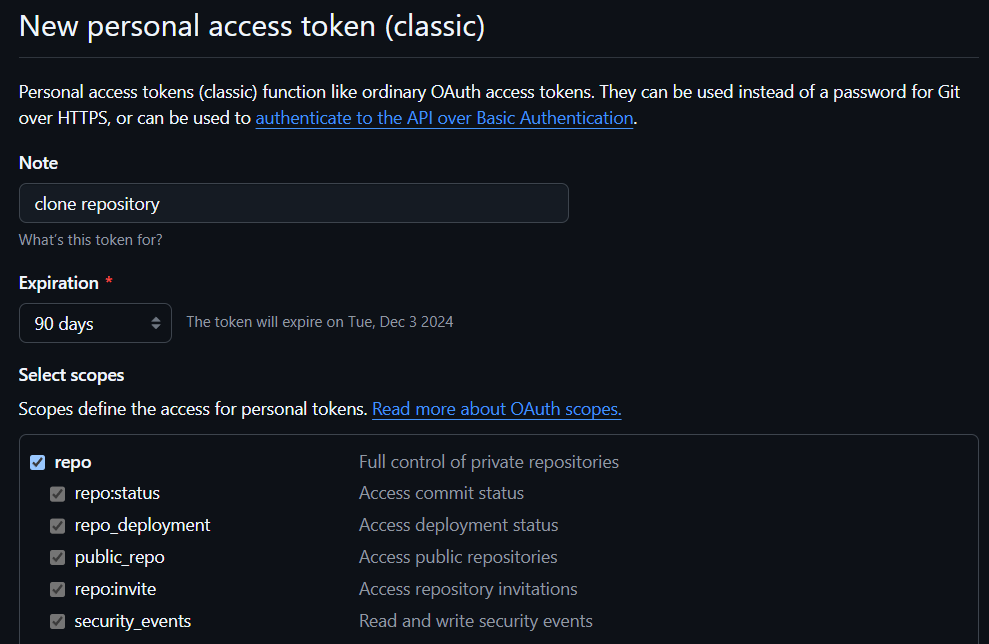
Go to GitHub and create a new repository.

I’ve named mine “application-codes-autorentify-project”

1. Download the required file from this link: <https://github.com/Silas-cloudspace/application-codes-autorentify-project>
2. Add it to the repository folder in your local machine
3. Open the repository “application-codes-autorentify-project” on VS code
4. Push it to GitHub “application-codes-autorentify-project”

**Create a personal access token on github**

1. This token will be used by docker to clone the application codes repository when we build our docker image
2. Github -> select your profile -> settings -> Developer settings -> Personal access tokens -> Tokens (classic) - > Generate new token -> Generate new token classic
3. Edit it as you see in the following example:



1. Remember to copy your personal access token and save it anywhere

**Create a Dockerfile**

1. Crete a new repository on GitHub and call it “docker-projects”
2. Open the repository docker-projects in VS Code
3. Create a new folder for the application.

* Run: mkdir rentzone
* cd rentzone
* touch Dockerfile

1. copy and paste into it the following: <https://github.com/Silas-cloudspace/docker-projects/blob/main/rentzone/Dockerfile>
2. We will use a LAMP stack to build this application. A lamp stack is a group of open-source software that we can use to build a dynamic web application:



* Linux is the operating system we will use to run the stack.
* Apache is the software we will use to serve the website.
* MySQL is the engine we will use to run our database.
* php is the programming language used to build web applications.

1. On line 93 we will replace the file named “AppServiceProvider.php” on our server.

We do this because we want to upload a value in it, so when we redirect our traffic from http to https our website function properly.

* Create the replacement file in our project folder
  + Copy “AppServiceProvider.php”
  + In VS code navigate to rentzone folder
  + Create a new file and name it “AppServiceProvider.php”
  + Paste into it the following: https://github.com/Silas-cloudspace/docker-projects/blob/main/rentzone/AppServiceProvider.php
  + This file is to redirect http traffic to https

**Create a script to build the Docker image**

1. Navigate to the “rentzone” directory in VS Code
2. Create a new file
   * Run: touch build\_image.ps1 (build\_image.sh for mac)
3. Copy and paste into it the following: <https://github.com/Silas-cloudspace/docker-projects/blob/main/rentzone/build_image.ps1>
4. Create a “.gitignore” file and add into it the “build\_images.ps1”

**Make the script executable**

Windows:

* Open powershell by running it as administrator in your computer
* Run the following command: Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

Mac:

* Run: chmod +x build\_image.sh

**Build the Docker Image**

1. Make sure you have Docker running on your machine
2. Navigate to rentzone directory on VS Code
3. Use powershell terminal and run:

.\build.image.ps1 (build\_image.sh for mac)

1. Run: docker image ls
2. Now we can see the image we just built

**Create a repository in Amazon ECR**

1. aws cli command to create an amazon ecr repository:
   * aws ecr create-repository --repository-name <repository-name> --region <region>
2. Navigate to rentzone directory on VS Code and run:
   * aws ecr create-repository --repository-name rentzone --region eu-west-2

**Push the Docker image to the Amazon ECR repository**

1. Retag docker image
   * Run: docker tag <image-tag> <repository-uri>

* You can check the tag name you gave to your image by running “docker image ls”. In this case you will see the name “rentzone”
* In order to get the repository uri, go to ECR in the AWS console

1. login to ecr

Run: aws ecr get-login-password | docker login --username AWS --password-stdin <aws\_account\_id>.dkr.ecr.<region>.amazonaws.com

1. push docker image to ecr repository

Run: docker push <repository-uri>

1. Run: docker tag rentzone 381491868231.dkr.ecr.eu-west-2.amazonaws.com/rentzone

If you run “docker image ls” you can check that the image was retagged succefully

1. Push the image into Amazon ECR

For this, first we need to login into ECR.

Run: aws ecr get-login-password | docker login --username AWS --password-stdin 381491868231.dkr.ecr.eu-west-2.amazonaws.com

1. Push the docker image into the ECR repository

Run: docker push 381491868231.dkr.ecr.eu-west-2.amazonaws.com/rentzone

**Create an environment file**

1. We will now create an environment file to store the environment variables we defined in our Dockerfile.
2. Open “docker-projects” directory in VS Code
3. cd rentzone
4. touch rentzone.env
5. Paste into it the following: <https://github.com/Silas-cloudspace/docker-projects/blob/main/rentzone/rentzone.env>
6. Replace it with your own information
7. Add “rentzone.env” to the .gitignore
8. git add .
9. git commit -m “created environment files”
10. git push

**Create Terraform Module for S3 bucket**

Navigate to the “terraform-modules” directory

mkdir s3

cd s3

touch main.tf variables.tf outputs.tf

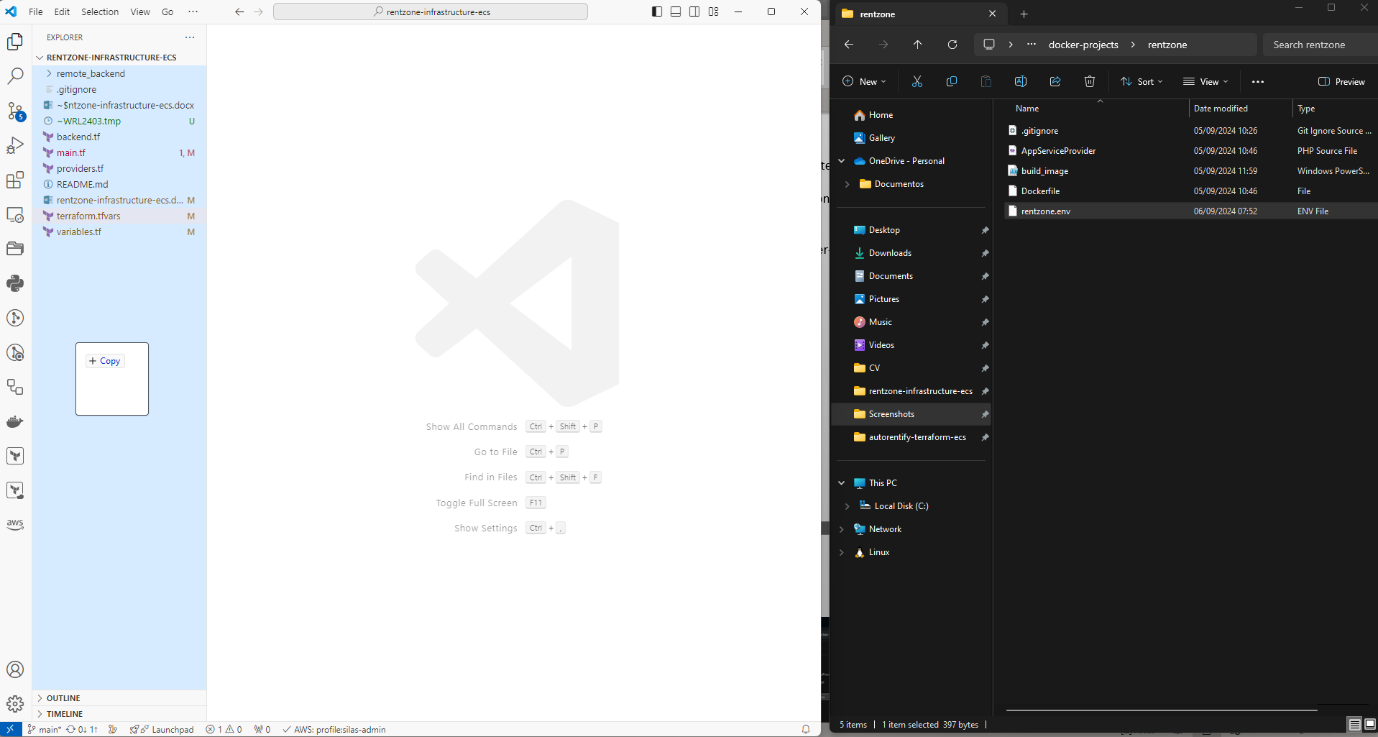
Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created s3 module”
* git push

**Create the S3 bucket**

1. This bucket will be used to upload the environment file we created.
2. When we create the ECS tags, the ECS tag will retrieve the environment variables from the file in the S3 bucket.
3. Copy the “rentzone.env” file stored in your computer on docker-projects > rentzone to “rentzone-infrastructure-ecs” directory in VS Code



1. Add “rentzone.env” to .gitignore

Navigate to the “rentzone-infrastructure-ecs” directory

Go to the main.tf file

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

Add //s3 to the end of the SSH URL because the module is located in the GitHub “s3” folder we recently created

**Important:** Your code will be bigger but run it only until the “module ‘s3\_bucket”. Comment the rest of it by:

* **For Windows/Linux**: Press Shift + Alt + A
* **For macOS**: Press Shift + Option + A

Run:

* terraform fmt
* terraform init
* terraform apply

**Create Terraform Module for the Task Execution Role**

Navigate to the “terraform-modules” directory

mkdir iam\_role

cd iam\_role

touch main.tf variables.tf outputs.tf

Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created iam\_role module”
* git push

**Create a Task Execution Role for your ECS Tasks**

Navigate to the “rentzone-infrastructure-ecs” directory

Go to the main.tf file

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

Add //iam\_role to the end of the SSH URL because the module is located in the GitHub “iam\_role” folder we recently created

**Important:** Your code will be bigger but run it only until the “module ‘ecs\_task\_execution\_role’”. Comment the rest of it by:

* **For Windows/Linux**: Press Shift + Alt + A
* **For macOS**: Press Shift + Option + A.

Run:

* terraform fmt
* terraform init
* terraform apply

**Create Terraform Module for ECS clusters, Task Definition and Service**

Navigate to the “terraform-modules” directory

mkdir ecs

cd ecs

touch main.tf variables.tf outputs.tf

Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created ecs module”
* git push

**Create an ECS Cluster Task Definition and Service**

Navigate to the “rentzone-infrastructure-ecs” directory

Go to the main.tf file

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

Add //ecs to the end of the SSH URL because the module is located in the GitHub “ecs” folder we recently created

**Important:** Your code will be bigger but run it only until the “module ‘ecs”. Comment the rest of it by:

* **For Windows/Linux**: Press Shift + Alt + A
* **For macOS**: Press Shift + Option + A

Run:

* terraform fmt
* terraform init
* terraform apply

**Create Terraform Module for ECS Auto Scaling Group**

Navigate to the “terraform-modules” directory

mkdir asg\_ecs

cd asg\_ecs

touch main.tf variables.tf

Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created asg\_ecs module”
* git push

**Create an ECS Auto Scaling Group**

Navigate to the “rentzone-infrastructure-ecs” directory

Go to the main.tf file

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

Add //asg\_ecs to the end of the SSH URL because the module is located in the GitHub “asg\_ecs” folder we recently created

**Important:** Your code will be bigger but run it only until the “module ‘asg\_ecs”. Comment the rest of it by:

* **For Windows/Linux**: Press Shift + Alt + A
* **For macOS**: Press Shift + Option + A

Run:

* terraform fmt
* terraform init
* terraform apply

**Create Terraform Module for Route53**

Navigate to the “terraform-modules” directory

mkdir route53

cd route53

touch main.tf variables.tf

Copy the terraform files from the shared repository into them.

Run:

* git add .
* git commit -m “created route53 module”
* git push

**Create Route53 DNS Records**

Navigate to the “rentzone-infrastructure-ecs” directory

Go to the main.tf file

Replace the source with the SSH URL of your terraform-modules repository in your GitHub account

Add //route53 to the end of the SSH URL because the module is located in the GitHub “route53” folder we recently created

Run:

* git add .
* git commit -m “updated files”
* git push
* terraform fmt
* terraform init
* terraform apply

ctrl+click on the url and you will see the deployed application on AWS using Terraform Modules and ECS Fargate.